

AD-A086 133

NOTRE DAME UNIV IN DEPT OF ELECTRICAL ENGINEERING
DESIGN AND IMPLEMENTATION OF A SPEECH CODING ALGORITHM AT 9600 --ETC(U)
APR 80 J L MELSA, D L CONN, A ARORA

F/8 17/8

DCA100-79-C-0005

NL

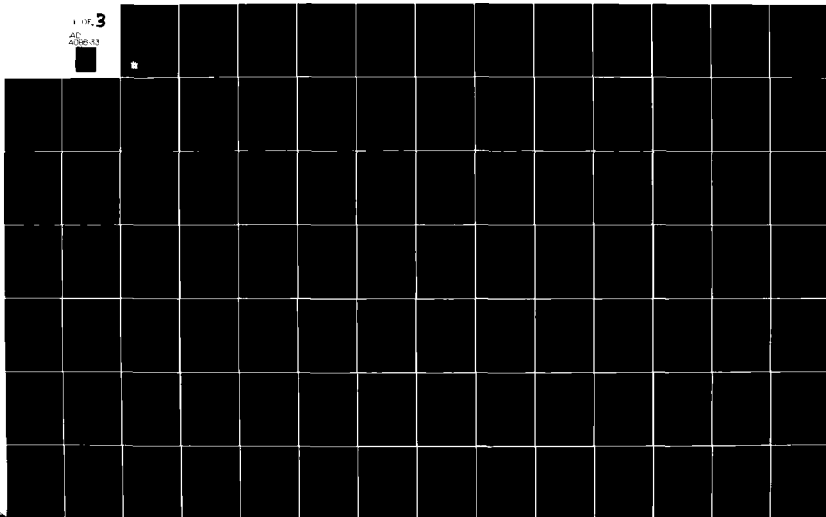
UNCLASSIFIED

1 OF 3

AD
2086-133



1



LEVEL

12

ADA 0861 33

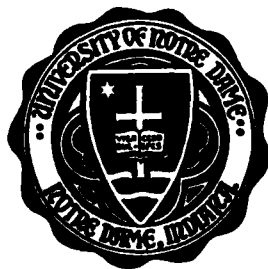
FINAL REPORT
VOLUME 1

DESIGN AND IMPLEMENTATION
OF A SPEECH CODING ALGORITHM
AT 9600 B/S

DTIC
ELECTE
JUL 1 1980
S C

DDC FILE COPY

This document has been approved
for public release and sale; its
distribution is unlimited.



Department of

ELECTRICAL ENGINEERING

UNIVERSITY OF NOTRE DAME, NOTRE DAME, INDIANA

80 6 30 189

12

FINAL REPORT
VOLUME 1

DESIGN AND IMPLEMENTATION
OF A SPEECH CODING ALGORITHM
AT 9600 B/S

STIC
ELECTE
JUL 1 1980
C

Prepared for

Defense Communications Agency
Defense Communications Engineering Center
1860 Wiehle Avenue
Reston, Virginia 22090

Contract No. DCA 100-79-C-0005

30 April 1980

This report is classified "Unclassified"
for public release and sale; its
distribution is unlimited.

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

| REPORT DOCUMENTATION PAGE | | READ INSTRUCTIONS BEFORE COMPLETING FORM |
|---|---|--|
| 1. REPORT NUMBER | 2. GOVT ACCESSION NO. <i>AD-A086 133</i> | 3. RECIPIENT'S CATALOG NUMBER |
| 4. TITLE (and Subtitle) Design and Implementation of a Speech Coding Algorithm at 9600 B/S | | 5. TYPE OF REPORT & PERIOD COVERED Final Report Nov. 1978 - April 1980 |
| 7. AUTHOR(s) James L. Melsa, et al. | | 6. PERFORMING ORG. REPORT NUMBER |
| 9. PERFORMING ORGANIZATION NAME AND ADDRESS Department of Electrical Engineering University of Notre Dame Notre Dame, IN 46556 | | 8. CONTRACT OR GRANT NUMBER(s) DCA 100-79-C-0005 |
| 11. CONTROLLING OFFICE NAME AND ADDRESS Defense Communications Agency Contract Management Division, Code 260 Washington, D.C. 20305 | | 10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS |
| 14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) | | 12. REPORT DATE April 1980 |
| | | 13. NUMBER OF PAGES 519 |
| | | 15. SECURITY CLASS. (of this report) Unclassified |
| | | 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE |
| 16. DISTRIBUTION STATEMENT (of this Report) Distribution of this document is unlimited. It may be released to the Clearinghouse, Department of Commerce, for sale to the general public. | | |
| 17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report) | | |
| 18. SUPPLEMENTARY NOTES | | |
| 19. KEY WORDS (Continue on reverse side if necessary and identify by block number) Speech coding, 9600 bps speech transmission, pitch extraction, adaptive residual coder, waveform reconstruction. | | |
| 20. ABSTRACT (Continue on reverse side if necessary and identify by block number) → This report describes a speech coding algorithm for digital transmission of speech at a rate of 9600 bits per second and the implementation of this algorithm on a speech processing system. The algorithm combines: Pitch extraction loop, Pitch compensating adaptive quantizer, Sequentially adaptive linear predictor, Adaptive source coding, → | | |

DD FORM 1 JAN 73 1473

EDITION OF 1 NOV 65 IS OBSOLETE

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)

to generate very high quality speech output. Although each of these elements has been previously applied to speech coding, the combination of all four of these elements has not been studied before. The speech coding algorithm has been implemented on a pair of CSPI MAP 300 Array Processors in real-time in the full-duplex mode.

This report has been bound in two volumes. The first volume contains the narrative description of the algorithm and its development and includes Chapters 1 through 11 and Appendices A through D of the report. The second volume describes the real-time MAP implementation and includes Chapters 12 and Appendices E through G.

| | |
|----------------|-----|
| Attention For | |
| WFO | Q&A |
| WFO | MAP |
| Unprocessed | |
| Justification | |
| By | |
| Date | |
| Priority Codes | |
| Mail and/or | |
| Special | |

A

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)

ABSTRACT

This report describes a speech coding algorithm for digital transmission of speech at a rate of 9600 bits per second and the implementation of this algorithm on a speech processing system. The algorithm combines

- Pitch extraction loop
- Pitch compensating adaptive quantizer
- Sequentially adaptive linear predictor
- Adaptive source coding

to generate very high quality speech output. Although each of these elements has been previously applied to speech coding, the combination of all four of these elements has not been studied before. The speech coding algorithm has been implemented on a pair of CSPI MAP 300 Array Processors in real-time in the full-duplex mode.

This report has been bound in two volumes. The first volume contains the narrative description of the algorithm and its development and includes Chapters 1 through 11 and Appendices A through D of the report. The second volume describes the real-time MAP implementation and includes Chapters 12 and Appendices E through G.

PROJECT PERSONNEL

Arvind Arora, research assistant

David L. Cohn, co-principal investigator

James M. Kresse, research assistant

James L. Melsa, principal investigator

Arun K. Pande, research assistant

Maw-lin Yeh, research assistant

FINAL REPORT

DCA CONTRACT 100-79-C-0005

| | <u>Page</u> |
|--|-------------|
| Abstract | i |
| Project Personnel | ii |
| 1. Introduction and Outline of Report | 1 |
| 1.1 Introduction | 1 |
| 1.2 Summary of Algorithm Requirements | 3 |
| 1.3 Outline of Report | 4 |
| 2. Algorithm Description | 5 |
| 2.1 Introduction | 5 |
| 2.2 Transmitter Buffer System | 11 |
| 2.3 Adaptive Low-Pass Filter | 16 |
| 2.4 Pitch Extraction | 18 |
| 2.5 Adaptive Residual Coder | 20 |
| 2.5.1 Adaptive Predictor | 20 |
| 2.5.2 Adaptive Quantizer | 22 |
| 2.6 Pitched Repetition | 26 |
| 2.7 Noiseless Source Coder | 28 |
| 2.8 Receiver | 34 |
| 3. Synchronization | 38 |
| 3.1 Sync. Algorithm Description | 40 |
| 3.1.1 Synchronization Acquisitions | 40 |
| 3.1.2 Synchronization Monitor | 43 |
| 4. Pitch Extraction Studies | 46 |
| 4.1 Introduction | 46 |
| 4.2 Pitch Extraction Algorithms | 47 |
| 4.3 Redundancy Removal | 54 |
| 4.4 References | 61 |
| 5. Tree Coding | 62 |
| 5.1 Introduction | 62 |
| 5.2 The (M,L) Algorithm | 63 |
| 5.2.1 Description | 63 |
| 5.2.2 Results | 65 |
| 5.3 Adaptive Tree Coding | 69 |
| 5.3.1 Description | 69 |
| 5.3.2 Results | 71 |
| 5.4 Conclusions and Suggestions for Further Research | 75 |
| 5.5 References | 77 |

| | | |
|--------|--|-----|
| 6. | Backward PARC | 78 |
| 6.1 | Introduction | 78 |
| 6.2 | System Structure | 80 |
| 6.3 | Performance Evaluation and Parametric Studies | 85 |
| 6.4 | Transmission Error Studies | 90 |
| 6.5 | Conclusions | 95 |
| 7. | Tandem Operation | 96 |
| 7.1 | Introduction | 96 |
| 7.2 | PARC in Tandem with CVSD | 97 |
| 7.3 | Effect of Background Noise on PARC Performance | 100 |
| 7.4 | References | 103 |
| 8. | Transmission Errors | 104 |
| 8.1 | Introduction | 104 |
| 8.2 | Simulation of Transmission Error | 105 |
| 8.3 | Minimizing Transmission Error Effects | 108 |
| 8.4 | Conclusion | 112 |
| 8.5 | Reference | 113 |
| 9. | Filtering | 114 |
| 9.1 | Introduction | 114 |
| 9.2 | Evaluation of Pre-emphasis | 115 |
| 9.3 | Filter Selection | 118 |
| 9.4 | Results | 121 |
| 9.5 | Low-Pass Filtering vs. Entropy | 123 |
| 9.6 | Results and Conclusions | 125 |
| 9.7 | References | 128 |
| 9.A | Derivation of Filter | 129 |
| 10. | Buffer Control | 136 |
| 10.1 | Introduction | 136 |
| 10.2 | Overflow control | 137 |
| 10.3 | Underflow control | 138 |
| 10.4 | Special considerations at the receiver | 139 |
| 10.5 | Conclusions and suggestions for further research | 140 |
| 11. | Source and Error Control Coding | 141 |
| 11.1 | Introduction | 141 |
| 11.2 | Source Coding | 142 |
| 11.2.1 | Quantizer levels | 142 |
| 11.2.2 | Side information | 144 |
| 11.3 | Error Control Coding | 145 |
| 11.4 | Conclusion and suggestions for further research | 146 |
| A. | FORTRAN Simulation of Algorithm | 147 |

| | | |
|--------|---|-----|
| B. | Segmented SNR Plots | 198 |
| B.1 | Introduction | 198 |
| B.2 | Listings | 199 |
| C. | Plotting Program | 242 |
| D. | PDP-11 D/A Programs | 247 |
| 12. | MAP Implementation | 263 |
| 12.1 | Introduction | 263 |
| 12.2 | PARC - Transmitter | 278 |
| 12.2.1 | The Pitch Computation Program | 281 |
| 12.2.2 | The Speech Digitization Program | 288 |
| 12.3 | PARC - Receiver | 295 |
| 12.4 | Noiseless Source Coder | 302 |
| 12.5 | Synchronizer, Decoder | 311 |
| 12.5.1 | Synchronization Acquisition Module | 313 |
| 12.5.2 | Decoder Module | 315 |
| 12.5.3 | Initialization Module | 318 |
| 12.5.4 | Decoder Constraints | 319 |
| 12.6 | Program Timing and Speed | 320 |
| E. | Resampling Program | 322 |
| E.1 | Operating Details | 325 |
| F. | CVSD Algorithm | 344 |
| F.1 | The CVSD System | 344 |
| F.2 | The Real-Time CVSD System | 346 |
| F.2.1 | Design Overview | 346 |
| F.2.2 | The Initialization Step | 348 |
| F.2.3 | The Processing Step | 351 |
| F.3 | Conclusion | 355 |
| F.4 | Reference | 355 |
| G. | Program Listings for Real Time Implementation of PARC on MAP-300 | 370 |

CHAPTER 1

INTRODUCTION AND OUTLINE OF REPORT

1.1 Introduction

This report describes the results of a sixteen month effort under DCA Contract 100-79-C-0005 to develop and implement a speech coding algorithm designed to produce very good quality speech at 9600 bits per second. The technique is based on the latest developments in speech digitization and is formulated to comply with the requirements described in the statement of work.

The method is based on techniques which have been reported in the literature but which are brought together here for the first time. The system combines elements of adaptive predictive coding and ADPCM systems and is known as PARC, Pitch extraction Adaptive Residual Coder. The pitch extraction loop used in adaptive predictive coding provided the input to a sequentially adaptive predictor, using backward coefficient adaptation which forms an estimate of the pitch-reduced signal. The error in this estimate is quantized by a pitch compensating adaptive quantizer. The resulting quantizer output is coded using an adaptive source coding procedure. The source code also permits transmission of pitch information and synchronization signals.

This algorithm was implemented on a pair of CSPI MAP 300 signal processors to generate a real-time, full-duplex speech encoding system.

This algorithm has several features which are significant in a full system application. The waveform reconstruction nature of the algorithm provides excellent performance in tandem with CVSD and in the presence of background noise. If bit rates higher than 9600 b/s are permitted,

the algorithm is easily adaptive to them. For example, a 16 kb/s version of the same algorithm will differ only in the number of quantization levels and the source code algorithm. It could easily be implemented using the same software.

1.2 Summary of Algorithm Requirements

The following requirements for the speech coding algorithm have been determined from the Statement of Work.

1. The speech processing system shall operate a transmission data rate of 9600 b/s.
2. The speech processing system shall produce very high quality speech reproduction. This requirement is interpreted to mean a signal-to-noise ratio of approximately 20 db.
3. The audio bandwidth of the speech coder shall be greater than or equal to 3200 Hz.
4. The speech coder shall produce good quality speech under conditions of a random transmission bit error rate of 1 percent.
5. The speech coder shall produce intelligible speech under conditions of acoustic background noise (60 db referenced to 20μ Newton/meter²) such as office noise.
6. The speech coder shall perform satisfactorily in tandem with a CVSD speech coder operating at a data rate of 16 kb/s. This tandem configuration shall provide speech intelligibility with minimal degradation compared with a single link of CVSD operating at 16 kb/s.

The algorithm shall be implemented on a pair of CSPI MAP 300 signal processor in real-time, full duplex mode with appropriate synchronization.

1.3 Outline of Report

Following the introductory material in this chapter, the next two chapters of the report describe the general details of the PARC algorithm. Chapter 2 describes the final form of the PARC speech digitization algorithm developed in this study.

The following nine chapters of the report describe in more depth the details of the algorithm and various studies that were made during this contract period but which do not necessarily appear in the final algorithm. Chapter 3 delineates the details of synchronization for full duplex operation. Chapter 4 describes various pitch extraction studies, while Chapter 5 describes details of the tree coding studies that were conducted. A special form of the PARC algorithm in which backward adaptation is used for the pitch extraction operation is described in Chapter 6. The operation of the algorithm in tandem with CVSD is discussed in Chapter 7, while Chapter 8 is concerned with transmission studies. The final algorithm uses an adaptive filter on the input speech to improve subjective performance. This algorithm is described in Chapter 9. Chapter 10 is concerned with the buffer control algorithms used in the PARC system, while Chapter 11 is concerned with source and error control coding for full-duplex operation.

Chapter 12 describes the real-time implementation of the algorithm on the MAP processor. The remainder of the report is a series of appendices which describe various details of the programming of the algorithms in both Fortran and on the MAP, as well as various support packages.

CHAPTER 2

ALGORITHM DESCRIPTION

2.1 Introduction

This chapter will describe the final form of the PARC speech digitization algorithm developed during this study. The algorithm will be decomposed into its constituent elements and each element will be described in turn. The underlying theory will be discussed and the details of the recommended implementation will be presented. Later chapters will describe the real-time implementation and the various studies which led to the recommended form of the algorithm.

The PARC digital speech communication system can be represented as shown in Fig. 2.1. The analog speech signal $s(t)$ is converted to a sequence of finely quantized samples $s(k)$. These quantized samples are stored in a buffer whose delay B_1 can vary over time. The PARC transmitter section does the actual data reduction to produce the quantizer level sequence. This is represented by the $q(k-B_1)$. This sequence, along with the side information quantized β and T , is noiselessly encoded into the bit stream $b(m)$ for transmission on the channel.

At the receiver end, the process is essentially reversed. The bit stream $b'(m)$ is decoded into the sequence $q'(k-B_1)$ and the side information β' and T' . The primes are used to allow for channel errors. The PARC receiver device converts this information into reconstructed speech $\hat{s}(k-B_1)$. This is buffered with a variable delay B_2 . A D/A output unit presents a filtered version of the delayed speech to the user. The overall system delay B_1+B_2 is a constant.

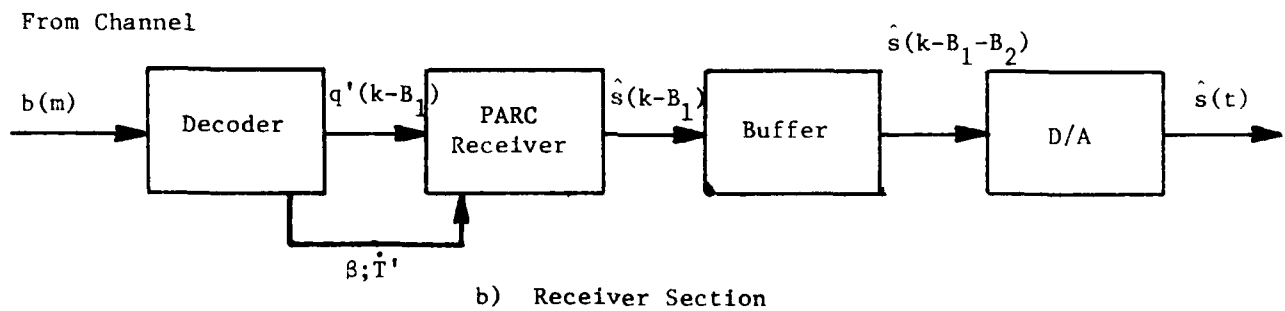
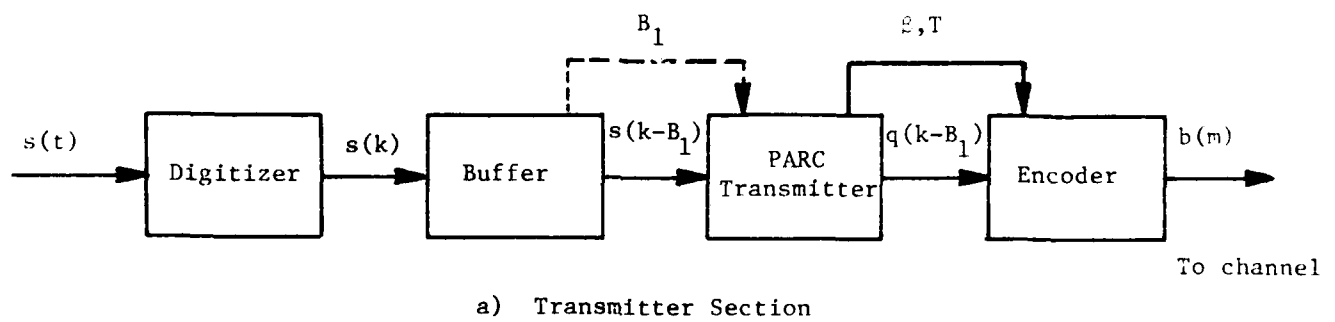


Fig. 2.1 Speech Digitization System

The analog filters were designed by GTE under a separate contract. They are described elsewhere. Therefore, the discussion here will begin after the continuous signal has been converted to $s(k)$.

Figure 2.2 shows a more detailed block diagram of the transmitter system. The system consists of the following major components:

- SAMPLE buffer
- Adaptive Low-pass filter
- Pitch extraction loop
- Adaptive Residual Coder
- Noiseless source coder.

As noted, the SAMPLE buffer receives the incoming speech samples and holds them for further processing. For notational simplicity, the samples at the output of this buffer will be referred to as $s(k)$. The adaptive low-pass filter is used as needed to help prevent an overflow of the SAMPLE buffer. The output of the adaptive low-pass filter $s_f(k)$ forms the input to the pitch extraction loop. The pitch extraction loop uses a block of these filtered samples $s_f(k)$ to estimate the pitch period T and the correlation coefficient β . Using this information, the pitch-reduced speech samples $v(k)$ are calculated by

$$v(k) = s_f(k) - \beta \hat{s}(k-T) \quad (2.1)$$

The pitch-reduced speech samples $v(k)$ are then processed by the Adaptive Residual Coder. An estimate $p(k)$ of $v(k)$, produced by the adaptive predictor is subtracted from $v(k)$ to form the prediction error $e(k)$. The prediction error $e(k)$ is then passed through an adaptive quantizer to yield the quantizer level $q(k)$. This quantizer level $q(k)$ is the

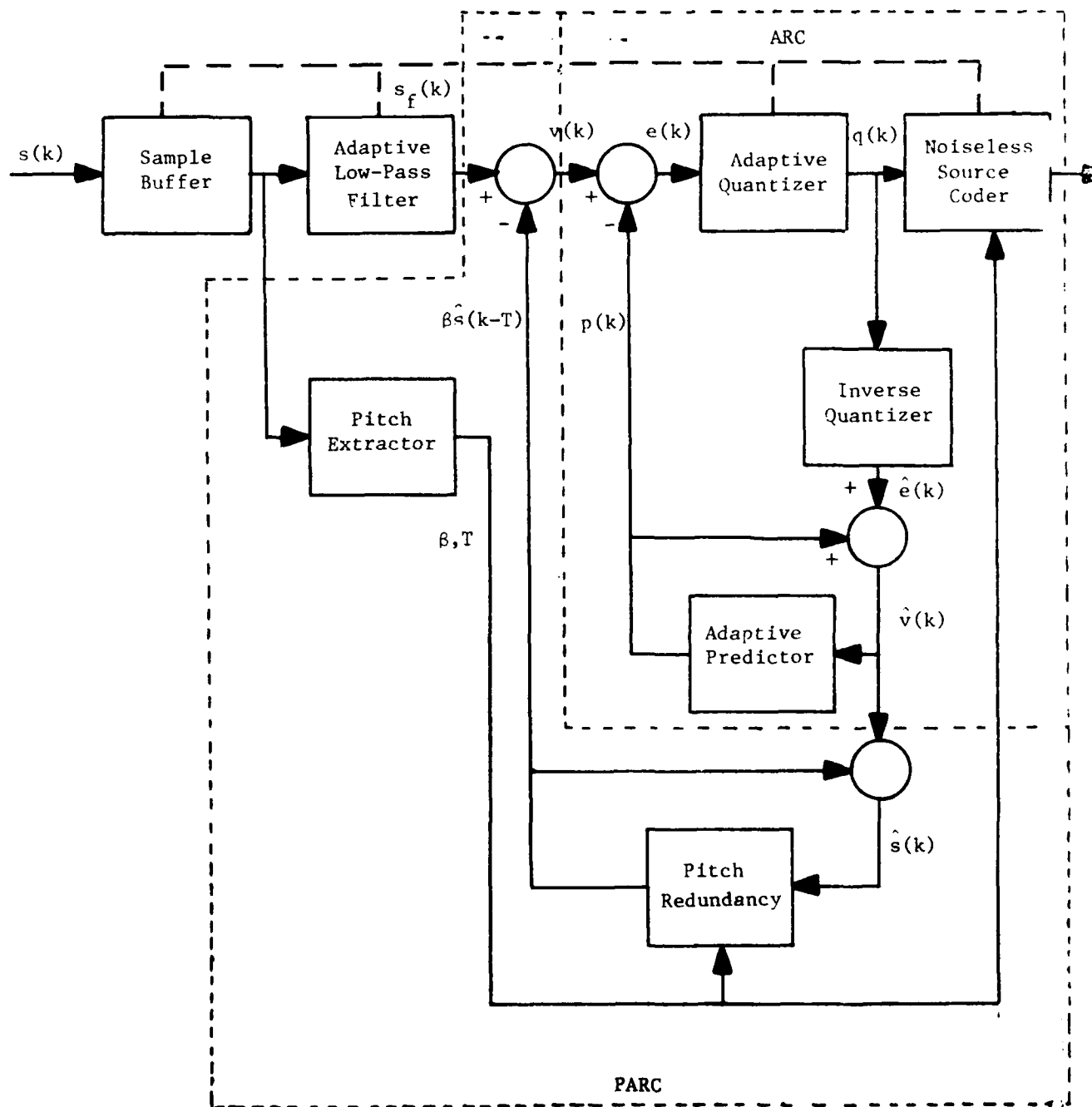


Fig. 2.2 PARC Transmitter

input to both the inverse quantizer (to update the rest of the system) and the noiseless source coder (to be transmitted down the channel). The noiseless source coder combines the quantizer level information $q(k)$, the pitch period T , and the correlation coefficient β , and generates the binary bit stream $b(m)$ to convey this information to the receiver.

The underlying design principle of the adaptation procedure is that all information used in updating the inverse quantizer and the predictor be available both at the transmitter and the receiver. This will allow the receiver to replicate these devices. Since the only information sent from the transmitter to the receiver is the quantizer output and pitching information, the adaptation procedures for the inverse quantizer and the predictor must use quantities derivable from them and a from pre-arranged initial state.

Although the PARC is basically a sequential system, the use of pitch redundancy reduction forces a block structure. For each block, new values of β and T are computed. The resulting block structure appears throughout the real-time implementation of the system.

Subsequent sections will describe each of the subsystems in the transmitter. The next section explains the operation of the transmitter buffer system. Since the noiseless source coder produces a variable number of bits for each sample, the rate at which samples are processed varies with time. Thus, the buffering operation is quite complex and important.

Section 2.3 describes the implementation of the adaptive low-pass filter. The filter is only activated when the SAMPLE buffer is almost full and it is only used to reduce the rate at which bits are generated. Thus, its operation is closely related to that of the SAMPLE buffer.

full and it is only used to reduce the rate at which bits are generated. Thus, its operation is closely related to that of the SAMPLE buffer.

The pitch extraction algorithm is a fairly standard AMDF system and is explained in Section 2.4. Although β and T are computed for a fixed number of speech samples, the number of samples they are actually used on will vary.

The adaptive residual coder is a sophisticated ADPCM system. Section 2.5 describes parametric modifications used to optimize the structure for pitch-reduced speech and to combat channel errors. A new feature, known as pitched repetition, has been added to the ARC to reduce the bit rate during voiced speech. It is described in Section 2.6.

A central feature of the PARC system is the noiseless source coding structure. It allows the most efficient use of all channel bits to yield the highest fidelity speech. In the final implementation, it includes the binary representation of $q(k)$, β and T as well as synchronization and error control. Section 2.7 gives the details of this component.

The next-to-last section of this chapter describes the structure of the receiver. It too has a complex buffer structure which must be considered during the system design. The final section is a list of references to the various techniques employed by PARC.

2.2 Transmitter Buffer System

The buffering of data flows in the transmitter is a fairly complex procedure. Due to the variable rate coding procedure, the overall delay experienced by a speech sample traveling through the transmitter varies with time. During unvoiced speech or silence, the delay will be short but during voiced speech, it can be over a thousand sample times. However, the receiver must produce one reconstructed speech sample for each sample that enters the transmitter. Thus, if there are no channel errors, the overall system delay is fixed.

There are actually four separate buffers used in the transmitter buffer system. Three are used to facilitate data transfer and to allow parallel processing; one accommodates the variable delay. The buffers are the ADAM buffer, the SAMPLE buffer, the LEVEL buffer and the BIT buffer. They are shown schematically in Fig. 2.3. The ADAM buffer, the LEVEL buffer, and the BIT buffer are each double buffers which allow parallel data processing and have essentially a fixed delay. The SAMPLE buffer, though, is a large circular buffer which accommodates the system's variable-rate encoding.

The operation of the four buffers is best described by first explaining the input and output for each of them. Continuous speech signal enters the PARC system through an analog speech interface. This interface conditions the signal for the Analog Data Acquisition Module (ADAM) by low-pass filtering it and by adjusting the signal level to the range of the analog-to-digital converter in the ADAM. The ADAM samples this signal at a rate of 6.4 Kss and places the samples into the ADAM buffer. Thus, the input to the buffer is a sequence of single speech samples at the rate of one each 156.25 μ sec.

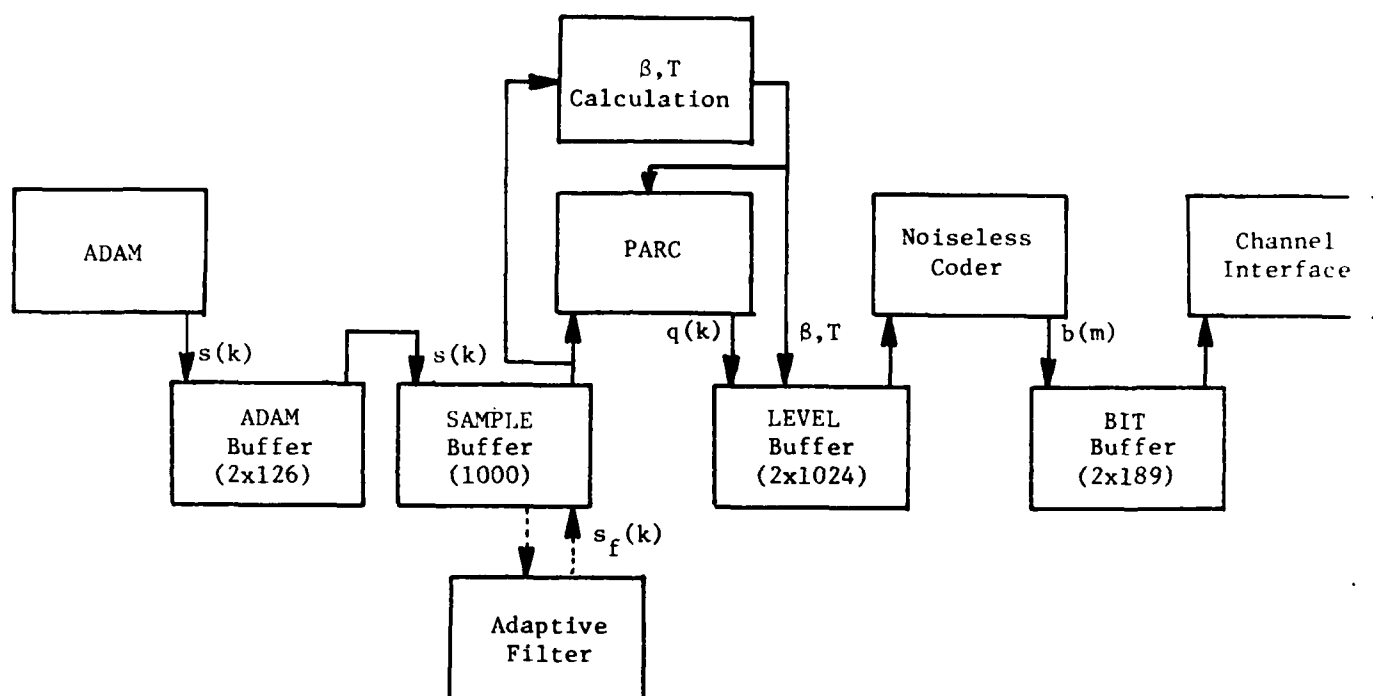


Fig. 2.3 Transmitter Buffer System

Data is removed from the ADAM buffer in blocks of 126 samples each 19 687.5 μ sec. Thus, data enters and leaves the buffer at an average rate of 6400 samples per second. The blocks of 126 samples are transferred to the SAMPLE buffer. The rate at which these samples are processed by the noise reducer and then are removed from the SAMPLE buffer depends on the number of bits they generate. The block-time of 19 687.5 μ sec. was selected to correspond to both 126 input sampling intervals and to the time allowed to transmit 189 bits on the channel. Thus, the number of samples N_B removed from the SAMPLE buffer in one block-time will be the number which generates at least 189 channel bits. This can range from as many as 500 or more during silence to fewer than 80 during voiced speech. The SAMPLE buffer must be large enough to accommodate this kind of variation without introducing undue delay.

The samples removed from the SAMPLE buffer in a given block time are processed by the PARC algorithm and the corresponding quantizer levels $q(k)$ are generated. The N_B samples will generate N_B quantizer levels. These are placed in the LEVEL buffer. Since PARC operates on a sample-by-sample basis, the $q(k)$ are loaded into their buffer one at a time.

The source coder generates full blocks of 189 bits. Therefore, it removes N_B quantizer levels from the LEVEL buffer at one time. The resulting block of bits are stored in the bit buffer. They are clocked out of this buffer at the rate of 9600 bits per second.

The ADAM buffer, as all double buffers, actually has two half-buffers, each of which holds 126 speech samples. While the ADAM is filling one half-buffer with the incoming samples, the other half-buffer can be emptied into the SAMPLE buffer. The SAMPLE buffer is a circular buffer which holds up to 1024 speech samples. There are two pointers associated with the buffer which keep track of where samples are to enter the buffer

and from where they are to leave the buffer. The distance between the pointers indicates how many samples are currently in the sample buffer.

Before the samples are removed from the SAMPLE buffer, three operations are performed. If the SAMPLE buffer is over a pitched repetition threshold, it will signal the transmitter to do the pitched repetition as described in Section 2.6. If the adaptive low-pass filter is called for, the oldest 80 samples in the buffer are filtered. The original samples are replaced with the filtered samples. The β and T values for de-pitching are then calculated based on the 80 oldest samples in the buffer. Samples are then passed to the PARC for processing and the corresponding quantizer levels are put into the LEVEL buffer. For each sample, the number of information bits required to represent the quantizer level is computed. The process will stop at either of the following three conditions:

1. The transmitter has generated the required 157 information bits.
2. The transmitter has no more input samples to process, i.e., the SAMPLE buffer is in an underflow condition. In this case, the transmitter will output a NULL code to the source encoder.
3. The transmitter has processed the maximum number of samples allowed to be in real-time.

The LEVEL buffer is a large double buffer capable of holding up to 1200 quantizer levels. Even though the buffer can hold 1200 levels, it will never have more than two blocks each of which generates 157 information bits. One half-buffer is used for incoming quantizer levels, while the other half-buffer is available to the noiseless source coder. The noiseless source coder takes these quantizer levels, and, with the associated quantized β and T, generates a 189-bit block which is placed in the BIT buffer.

The BIT buffer is a double buffer which holds 378 bits. One half-buffer receives bits from the channel coder, while the other half-buffer is available for output to the channel. Output takes place through the Input/Output Scroll (IOS-2). The IOS-2 provides the bits to the modem interface contained in the speech interface unit.

2.3 Adaptive Low-Pass Filter

Adaptive low-pass filtering is used in the system to provide a soft-failure capability under certain circumstances. When the SAMPLE buffer fills more rapidly than it is being emptied, for example during voiced speech, it is possible that it could overflow. It was found that low-pass filtering the speech mitigated this problem. When the SAMPLE buffer is nearly full, therefore, the speech is low-pass filtered to help guard against buffer overflow.

The recommended low-pass filter has been derived from a first order Butterworth filter, using the bilinear transformation to obtain a digital filter. The general form of the transfer function of this type of filter is

$$H(z) = \frac{\omega_{CA} + \omega_{CA} z^{-1}}{(\omega_{CA} + 1) + (\omega_{CA} - 1)z^{-1}} \quad (2.2)$$

where

$$\omega_{CA} = \tan \frac{\pi f_c}{f_s}$$

f_c = cutoff frequency in Hz.

f_s = sampling frequency in Hz.

The selected parameters are $f_c = 1800\text{Hz}$ and $f_s = 6400\text{ Hz}$. Transforming back into the sample domain, the filter equation can be written as

$$s_f(k) = As(k) + As(k-1) + Bs_f(k-1) \quad (2.3)$$

where

$$A = \frac{CA}{CA + 1}$$

$$B = \frac{CA - 1}{CA + 1}$$

The adaptive low-pass filtering operates by checking the number of samples in the SAMPLE buffer just prior to the pitch extraction calculations. If there are fewer than 501 samples in the SAMPLE buffer at that time, the low-pass filtering is skipped. If there are 501 or more samples in the same buffer, the filtering is implemented. Thus, filtering is only used when buffer overflow is threatened.

When filtering is called for, a block of 80 samples are low-pass filtered and the original samples are replaced by the filtered samples. A block of N_B samples are then processed by PARC. Note that N_B may be less than 80 so that some filtered samples may remain in the buffer and may be filtered again during the next block time.

The filtering is performed in blocks of 80 for two reasons. First, filtering in large blocks reduces the number of transitions between filtered and unfiltered speech. Second, the larger block causes some samples to be multiple-filtered if the sample buffer continues to fill. The design of the filter causes the effect of this multiple-filtering to be similar to filtering with a lower cutoff frequency. For the recommended parameters, using the 1800 Hz cutoff frequency filter twice results in an overall filter with a cutoff frequency of about 1350 Hz. In this way, the low-pass filtering is automatically increased when needed.

2.4 Pitch Extraction

It is well known that voiced speech is highly correlated from pitch period to pitch period. A long-term prediction of $s_f(k)$ given by

$$s(k|k-T) = \beta s_f(k-T) \quad (2.4)$$

can be a good approximation to $s_f(k)$ for proper choice of β and T . The optimum scale factor β depends on the correlation between $s_f(k)$ and $s_f(k-T)$, and the best T is an estimate of the pitch period measured in samples. The goal in selecting β and T is to minimize the time average prediction error over a block of K samples

$$E = \frac{1}{K} \sum_{j=1}^K [s_f(j) - \beta s_f(j-T)]^2 \quad (2.5)$$

In most applications, β and T are computed and used over a given block of K samples. This procedure is modified in PARC. New values of β and T are computed each block-time. Therefore, they are held constant for N_B samples and, it will be recalled, N_B varies from block to block. However, it is not possible to determine N_B until after β and T have been chosen. Therefore, β and T are always calculated for a fixed block size and used for a variable number of samples. In the recommended implementation, the fixed computation block is $K = 80$.

The pitch period estimate T is computed by forming the Average Magnitude Difference Function (AMDF) and picking the value of T for which this function is minimized. The AMDF function is

$$A(T) = \sum_{j=1}^K |s_f(j) - s_f(j-T)| \quad (2.6)$$

In this way, the value of T selected usually matches the value obtained by minimizing the error E in Eq. (2.5) but with far fewer computations. Once T has been found, the error E is minimized by selecting β according to

$$\beta = \frac{\sum_{j=1}^K s_f(j)s_f(j-T)}{\sum_{j=1}^K s_f^2(j-T)} \quad (2.7)$$

For some blocks of speech, such as those including transition regions from silence or unvoiced speech to voiced speech, the value of β given by Eq. (2.7) can be large. The use of such large values, however, can actually decrease the overall performance. Therefore, β was limited to the range of $[-2, 2]$. The β 's were uniformly quantized over this range. It was found that system performance was relatively insensitive to quantization noise on β . Therefore, β is quantized to 97 levels. This is represented in the transmitted block with seven bits. The 31 patterns of these seven bits which do not represent valid β values are used for another purpose.

By extracting pitch from different sentences and different speakers, it was found that the pitch period T varied between 24 and 70 samples. Hence, the searching range was chosen to be between 20 and 83 yielding 64 possible values of T , which requires an 8-bit codeword for T .

2.5 Adaptive Residual Coder

The heart of the PARC system is an Adaptive Residual Coder (ARC). The version recommended here has been modified to optimize its operation as a part of PARC. The input to the ARC is the pitch-reduced speech.

$$v(k) = s_f(k) - \hat{\beta}s(k-T) \quad (2.10)$$

where $\hat{s}(k-T)$ is the reconstructed version of $s_f(k-T)$ available at both the transmitter and receiver. The ARC consists of two principal sub-systems: an adaptive predictor and an adaptive quantizer. These will be described in separate subsections.

2.5.1 Adaptive Predictor

The adaptive predictor produces a linear prediction $p(k)$ given by

$$p(k) = \sum_{i=1}^N a_i(k) \hat{v}(k-i) \quad (2.11)$$

which is to be an estimate of $v(k)$. The $\hat{v}(k-i)$ are the receiver's estimate of $v(k-i)$. It can be argued that the predictor order N should match the order of the system which generates the $v(k)$. However, predictors of order larger than 4 yield unsatisfactory performance in the presence of channel errors. Therefore, $N=4$ is used.

If the $a_i(k)$ accurately model the $v(k)$, and if the $\hat{v}(k-i)$ are close to the $v(k-i)$, then $p(k)$ will be a good approximation to $v(k)$. The $a_i(k)$ are adaptive, and after $p(k)$ is formed, they are updated. They are adapted according to steepest descent of $e^2(k)$. This is approximated in the system by the following updating algorithm:

$$a_i(k+1) = \delta b_i + (1-\delta) \left[a_i(k) + \frac{g \hat{v}(k-i) \hat{e}(k)}{\langle |\hat{v}(k)| \rangle^2} \right] \quad (2.12)$$

where $\langle |\hat{v}(k)| \rangle$ is a biased exponential time average of $|\hat{v}(k)|$

$$\langle |\hat{v}(k)| \rangle = (1-\alpha) \sum_{j=0}^{\infty} \alpha^j |\hat{v}(k-j)| + \text{RMSMIN}$$

Thus, the $a_i(k)$ updating algorithm has eight parameters: δ , g , α , RMSMIN and b_i for $i=1,2,3$ and 4. Three of them, δ , g and α , essentially determine how much memory there is in the updating process. In order to minimize the effect of channel errors, the memory time was reduced from what would be optimal in the error-free case. This did not significantly degrade performance. The recommended values of these parameters are

$$\delta = 0.01$$

$$g = 0.02$$

$$\alpha = 0.90$$

The parameters b_i represent the quiescent values of the coefficients $a_i(k)$. The values used in the original ARC are also recommended here.

$$b_i = \begin{cases} 0.7 & i=1 \\ 0 & i=2, 3 \text{ or } 4 \end{cases}$$

The quantity RMSMIN is perhaps the most sensitive parameter in the algorithm. It determines the minimum value of $\langle |\hat{v}(k)| \rangle$ which affects both the adaptive predictor and the adaptive quantizer. The lower RMSMIN, the more the system responds during low level signals. This reduces granular noise and increases the data rate. The higher data rate means that the sample buffer fills faster leading to more low-pass filtering and increased use of pitched repetition. The value selected for RMSMIN must be matched to the dynamic range of $s(k)$. When $s(k)$ is represented on the interval $(-2048, 2047)$, an RMSMIN of

$$\text{RMSMIN} = 50$$

produces a good tradeoff.

2.5.2 Adaptive Quantizer

The prediction error $e(k)$ is the input to the adaptive quantizer whose basic design is illustrated in Fig. 2.4. The input is normalized by an adaptive scaling factor $\sigma(k)$ and the result is compared to a set of thresholds T_i . The recommended thresholds are symmetric and are illustrated in Fig. 2.5 and listed in Table 2.1. The level in which the normalized input falls specifies the quantizer output $q(k)$. The inverse quantizer output $\hat{e}(k)$ is the quantized version of the quantizer input. It is the product of a scaling factor $f(q(k))$ and the state variable $\sigma(k)$. The recommended scale factors are tabulated in Table 2.2. The recommended thresholds were computed to be equidistant between the scaling factors.

The state variable $\sigma(k)$ is designed to be an approximation to the standard deviation of $e(k)$. Most of the time the scaled average of $|\hat{v}(k)|$ is an acceptable estimate. However, in voiced speech at the beginning of a pitch period, $e(k)$ is much larger than usual. Therefore, whenever one of the outermost quantizer level occurs, $\sigma(k)$ is significantly increased. If no further outer level occurs, $\sigma(k)$ decays back to the scaled average of $|\hat{v}(k)|$. Thus, $\sigma(k)$ is updated by

$$\sigma(k) = \max\{SMIN < |\hat{v}(k)| >, \phi[q(k)]\sigma(k-1)\} \quad (2.13)$$

The first term in the braces of Eq. (2.13) usually dominates. This means that the quantizer behavior is largely determined by $SMIN < |\hat{v}(k)| >$ and, hence, by the product of $SMIN$ and $RMSMIN$. It is recommended that the scale factor $SMIN$ be set to 0.3.

The second term in the braces only affects performance at the beginning of pitch periods. The quantizer expansion factors $\phi[q(k)]$ are given in Table 2.2.

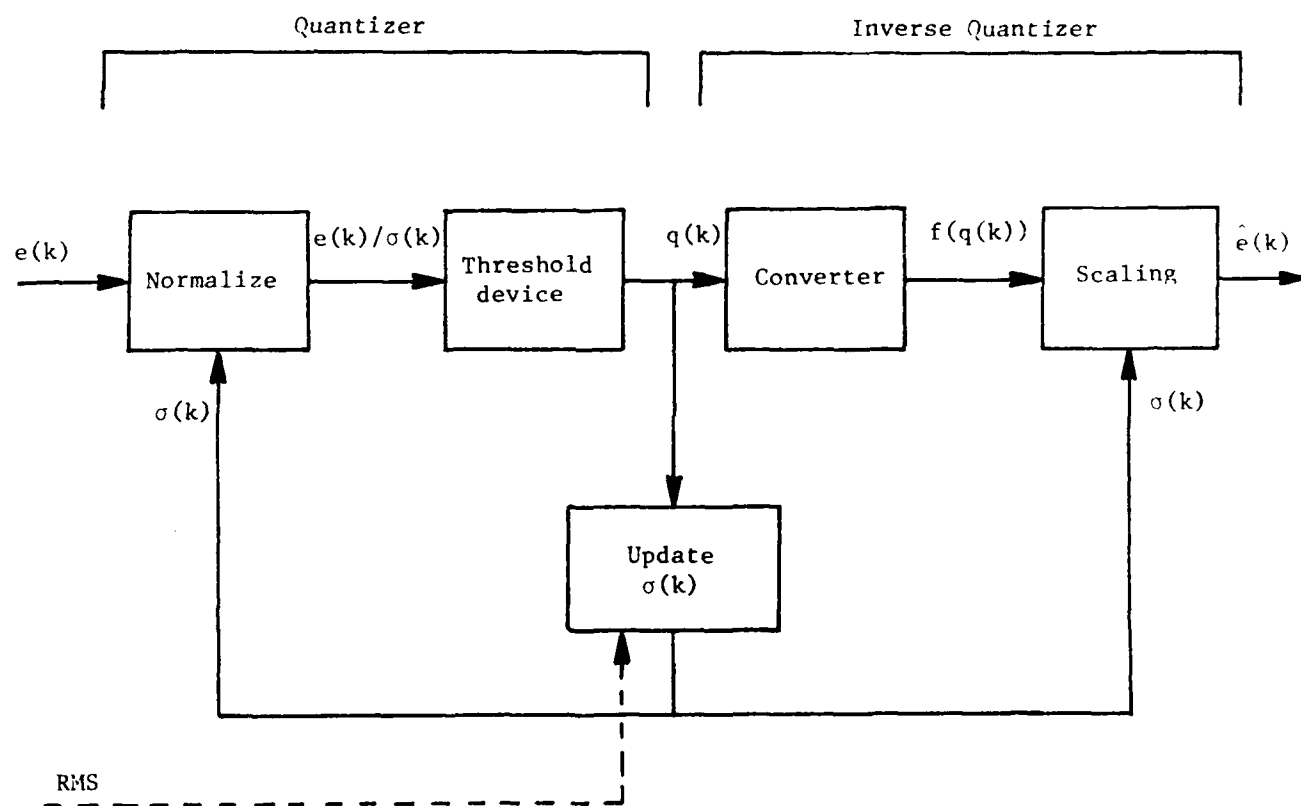


Fig. 2.4 Adaptive Quantizer

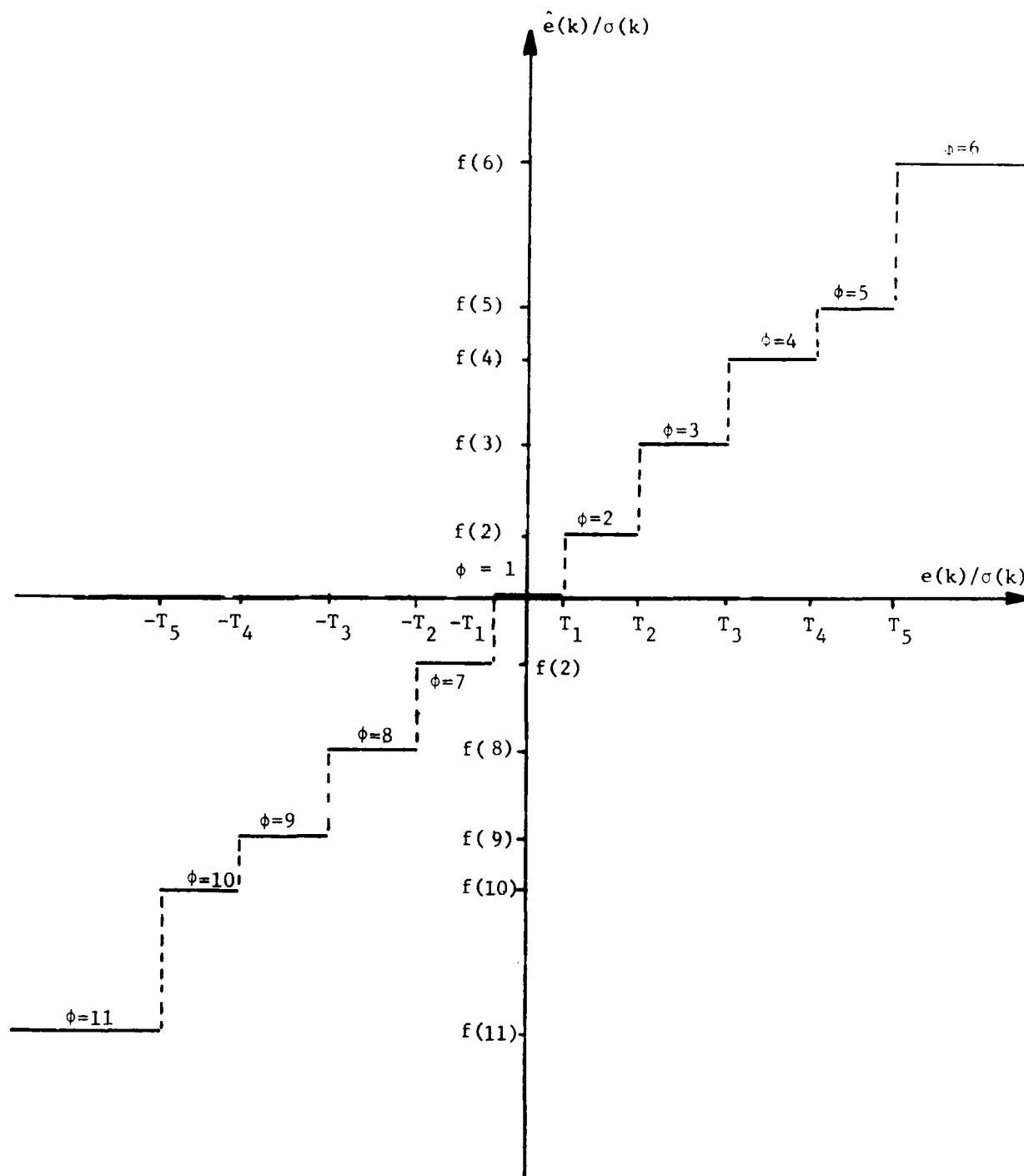


Fig. 2.5 Quantizer Thresholds and Output Levels

Table 2.1 Quantizer Thresholds

| i | T_i |
|-----|-------|
| 1 | 0.90 |
| 2 | 3.03 |
| 3 | 5.38 |
| 4 | 7.75 |
| 5 | 10 |

Table 2.2 Quantizer Scaling Factors and Expansion Factors

| $q(k)$ | $f[q(k)]$ | $\phi[q(k)]$ |
|--------|-------------|--------------|
| 1 | 0.00 | 0.7 |
| 2,7 | ± 1.80 | 0.8 |
| 3,8 | ± 4.25 | 0.9 |
| 4,9 | ± 6.50 | 1.1 |
| 5,10 | ± 8.00 | 1.5 |
| 6,11 | ± 12.00 | 2.2 |

2.6 Pitched Repetition

Even with adaptive low-pass filtering, sections of voiced speech can generate a large number of bits rapidly. This could cause the SAMPLE buffer to overflow. To avoid the overflow problem, the system uses a technique known as Pitched Repetition. When overflow is imminent, a block of samples is deleted at the transmitter and replaced at the receiver with previous reconstructed speech. The replacement samples are the reconstructed samples from the previous pitch period. In order to have an absolute control on the overflow condition, the repetition size has to be carefully set. During a voiced region, the bit rate may be as high as 3.4 bits per sample. Thus, the repetition size of 80 samples is enough to force more than 126 samples to be transmitted in a time slot, i.e., the total number of output samples is more than that of input samples. The decision to use pitched repetition is made at the beginning of each block and a special signal is used to alert the receiver.

If the SAMPLE buffer contains more than 850 samples at the beginning of a block-time, pitched repetition is employed. First, the pitch period T is computed in the usual way. The output pointer in the SAMPLE buffer is then moved forward 80 samples. Thus, these 80 samples will not be processed by PARC. They are, however, involved in the β calculation which takes place with the output pointer at its new location.

The receiver must still produce an output for those samples which are not processed by PARC. It does this by using previous outputs delayed by one pitch period. Thus, if the first sample skipped is $s_f(k)$, it is represented at the receiver by $\hat{s}(k-T)$. Similarly, $s_f(k+1)$ is represented by $\hat{s}(k+1-T)$ and so on. The transmitter must know this since it uses $\hat{s}(k)$ values in computing $v(k)$. Therefore, the $\hat{s}(k)$ buffer is filled

with prior $\hat{s}(k-T)$ values as part of the pitched repetition. After this, PARC resumes normal operation.

This method of repeating short intervals of samples from the previous pitch period works with little distortion during voiced speech because samples one pitch period apart are highly correlated. In this way, the buffer overflow problem is overcome with a minimum of additional distortion.

The receiver must be informed that a period of pitched repetition is taking place. As detailed in the next section, this is accomplished through use of the unused values of β .

2.7 Noiseless Source Coder

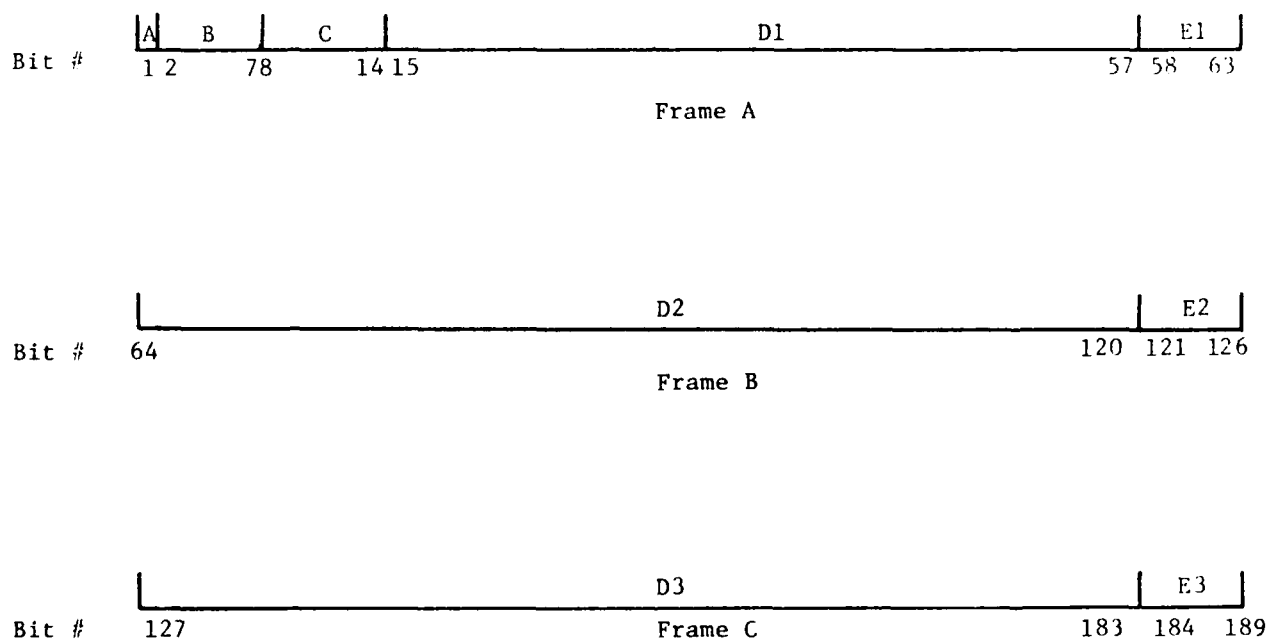
The function of the noiseless source coder is to combine all of the information from PARC and produce the corresponding bit stream for transmission. The output of the coder are blocks of 189 bits. Each block represents one set of values of β and T and N_B level variables. In addition, the block provides for synchronization and error control.

The format of a typical block is illustrated in Fig. 2.6. The block is divided into three 63-bit frames to facilitate error control. The last six bits of each frame, denoted E_1 , E_2 and E_3 in Fig. 2.6, are used for error correction. A single-error-correcting (57,63) Hamming code is used. The first 57 bits in each frame are available for information. The (57,63) codes can each correct any single error so up to three errors per block can be corrected.

The first bit in the block, shown as Field A, is for block synchronization. It is set to 0 in odd numbered blocks and 1 in even numbered blocks. The receiver detects this pattern and knows where the block begins.

The next field in the block is six bits long and contains information specifying on the pitch period T . The pitch period is constrained to be an integer between 20 and 83, so six bits can transmit the pitch period without quantization error.

Field C in a normal frame is seven bits long and contains information on the pitch correlation coefficient β . It was found that β can be quantized fairly coarsely with negligible degradation, so that 97 possible values in the range of $[-2,2]$ are allowed. This means that 97 of the possible 128 values of Field C are used to represent β . If one of the other 31 patterns appears, it indicates that this is not a normal block. Rather, it is one using pitched repetition. The pitched repetition blocks are discussed later.



| Field | Content |
|----------|---------------------------------------|
| A | Synchronization Bit |
| B | Pitch Period T |
| C | Pitch correlation coefficient β |
| D1,D2,D3 | Quantizer levels |
| E1,E2,E3 | Parity bits |

Fig. 2.6 Normal Block Format

Following the β field in a normal block are the 157 bits representing the quantizer levels. These are denoted as Fields D1, D2 and D3. The source code used for the quantizer levels are described in Table 2.3. Quantizer levels 2 through 11 are each represented by a variable length bit pattern. Quantizer level 1, however, occurs so often that there are two ways of representing it. Isolated occurrences are represented by the 1-bit sequence 0. If level 1 occurs 14 times in a row, the entire string is represented by the sequence 1110. Thus, the source code is an over-full variable-length to variable-length mapping.

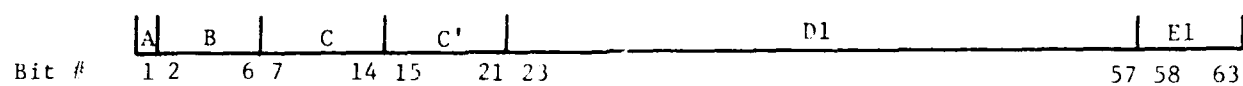
There is also a bit pattern associated with the null quantizer level sequence; this is used to fill out a block when the samples in the sample buffer do not generate at least 157 bits. Because of the variable number of bits used for different quantizer level sequences, an integral number of samples will not always generate exactly 157 bits. If there are more than 157 bits generated by a set of samples, the excess bits are the first bits transmitted in the next block's quantizer level field.

The normal format described above is used under most circumstances. If pitched repetition block is to be signaled to the receiver, however, the block format is changed slightly in the first frame of 63 bits, as shown in Fig. 2.7. Pitched repetition is signaled by using a special bit pattern, a "false β ", in the field C which is usually reserved for β . The next 7 bits are then taken from the quantizer level field D1 to create Field C' which transmits the actual β . Thus, in such a block, only 150 bits are used for quantizer levels.

To protect against missing a "false β ", or deciding one in error, the bit patterns for the β were carefully selected. They were designed so that neither situation can occur due to a single bit error. The "false β "

Table 2.3 Description of Quantizer
Level Source Code

| Quantizer Level Sequence | Bit Sequence |
|-----------------------------|-----------------|
| 1 | 0 |
| 7 | 100 |
| 2 | 101 |
| 8 | 1100 |
| 3 | 1101 |
| 14 1's | 1110 |
| 9 | 111100 |
| 4 | 111101 |
| 10 | 1111100 |
| 5 | 1111101 |
| 11 | 11111100 |
| 6 | 11111101 |
| null | 11111110 |



| Field | Content |
|-------|---------------------------------------|
| A | Synchronization bit |
| B | Pitch period T |
| C | False β |
| C' | Pitch correlation coefficient β |
| D | Quantizer levels |
| E1 | Parity bits |

Fig. 2.7 First Frame During Pitched Repetition

is represented by 0000000. The 7 patterns with a single one and the 21 patterns with two ones are never transmitted. Thus, only 99 patterns are available for true β values. At the receiver, if field C contains either all zeros or a single 1, it is interpreted as the "false β ". If no more than one channel error has occurred, this will happen if and only if a "false β " was actually transmitted.

2.8 Receiver

The principal elements of the receiver are illustrated in Fig. 2.8. The received bit sequence $b(m)$ is monitored by a synchronizer which establishes the beginning of a block. The decoder transforms the bits in a block into the quantizers levels $q(k)$ and the pitch parameters β and T . These are then processed by the inverse of the PARC algorithm. The $\hat{s}(k)$ values are stored in a variable delay receiver buffer. They are clocked out of the buffer and interpolated to produce the un-sampled output signal \hat{s}_{out} . The primes on all of these quantities introduced earlier to account for possible channel errors have been dropped for notational simplicity.

The synchronization actually operates in two modes: initial establishment and monitoring. During the establishment phase, the rest of the receiver system is disabled. The synchronizer looks for a sequence of bits spaced by 189 bit-times whose polarity oscillates. When a sequence of 10 bits with perfect oscillation is found, the synchronizer decides that it must represent the sync bit in 10 successive 189-bit blocks. The rest of the receiver is enabled and the synchronizer changes to the monitor mode. If it detects a significant number of errors in the sync bits, it assumes block synchronization has been lost. The receiver is disabled and the synchronizer returns to the synchronization mode.

Once synchronization has been established, the decoder can go to work. It basically inverts the operations performed by the noiseless source coder. Thus, it works on a full block of bits. The Hamming codes are decoded and any correctable bit errors are corrected. The values of β and T are found and the sequence of quantizer level values is formed.

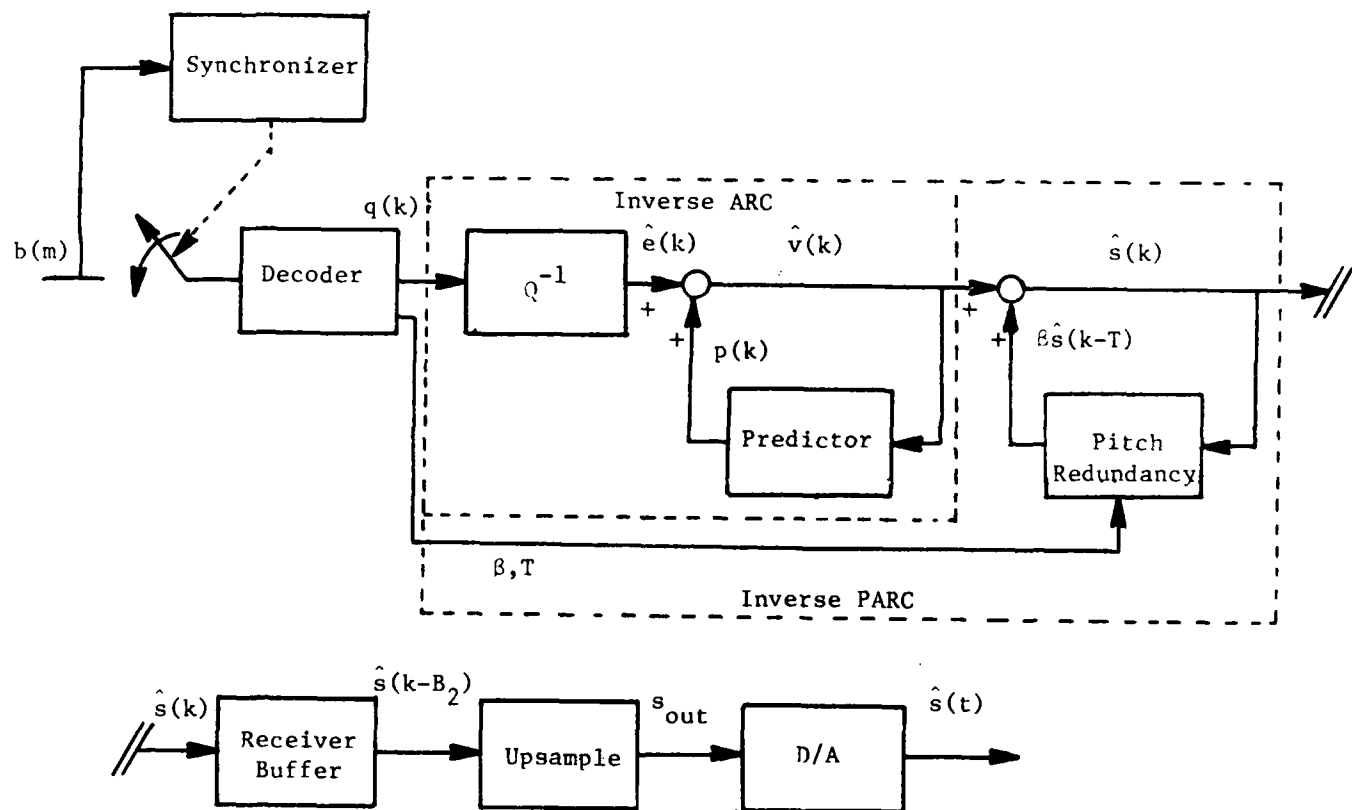


Fig. 2.8 Receiver Structure

If pitched repetition is being used, a special code is set. All of this information is placed in a double-buffer which interfaces with the PARC receiver.

The PARC receiver processes a full block of data. The B and T values are fixed for the block but the number of samples handled is the variable N_B . If pitched repetition is called for, the old $\hat{s}(k)$ values are produced before the ARC receiver is enabled.

The output $\hat{s}(k)$ from the PARC receiver are stored in the circular receiver buffer. This buffer complements the variable delay transmitter buffer and requires a carefully designed control algorithm. If there are no channel errors, the total delay for the system will be a constant. Therefore, the sum of the transmitter delay B_1 and the receiver delay B_2 will be fixed.

The buffer control logic in the decoder is designed to prevent the received sample buffer from ever overflowing or underflowing. In normal operation neither of those can happen but channel errors can add or delete samples. Since samples are removed from the buffer at a known rate, the buffer control logic will always know how many samples are in the buffer. If there is not enough room in the receiver buffer for all of the $\hat{s}(k)$ in a block, the excess $q(k)$ are simply discarded. Since a full receive buffer corresponds to an empty transmitter buffer, this usually occurs during silence. The deletion of silence is generally not a problem.

As the transmitter buffer fills, the receiver buffer empties. If channel errors have caused the deletion of samples, it is possible for the receiver to run out of $\hat{s}(k)$ values. This underflow condition is also prevented by the buffer control logic. If the number of $\hat{s}(k)$ values in a block plus the number of $\hat{s}(k)$ stored in the buffer do not total at least 126,

additional $q(k)$ representing silence are added to the LEVEL buffer. Thus, there are always enough.

The upsampling has been added to reduce the sampling noise caused by the non-ideal nature of the analog output filter. The 6400 samp/sec sampling rate is at the Nyquist rate of the 3200 Hz output filter. Thus, severe aliasing is possible. The upsampler effectively increases the sampling rate to 12800 samp/sec. It does this by interpolating between successive $\hat{s}(k)$ values. It was found that linear interpolation was sufficiently accurate to greatly reduce the aliasing.

CHAPTER 3

SYNCHRONIZATION

Although PARC is basically a sequential algorithm, the use of pitch redundancy reduction and error control forces on it a block structure. The transmitter quantizes a block of N_B speech samples using a set of pitched reduction parameters and encodes this into frames of binary information. At the other end, the receiver must identify these frames to be able to properly decode the information it receives. This necessitates some sort of synchronization between the transmitter and the receiver. In this chapter, the synchronization technique used in PARC and its implementation is described. Further, the operation is analyzed to illustrate its satisfactory performance.

There are two aspects to the synchronization operation performed in the receiver. The receiver must first locate the frame boundaries in the received bit stream. This is called synchronization acquisition. After acquisition, it must monitor the frame boundaries on a continuing basis to ensure that sync is not lost. This is called synchronization monitor. Contract requirements specify that bits are not dropped during transmission. Therefore, once synchronization is properly acquired, there should be no way of losing it. However, there are several reasons for providing the sync monitor. Sync acquisition is a probabilistic operation; and although there is a high probability of acquiring sync properly in one attempt, in case of a wrong decision, the sync monitor provides a way for re-attempting sync acquisition. There are other abnormal conditions that inevitably occur during algorithm development and testing which can also cause sync to be lost. Some of these are bad connections on the digital I/O connectors, faulty

IOS operation (see Chapter 12), reinitialization of the algorithm at one end of the communication system. All these make it imperative to provide the algorithm with the sync monitor, in other words, with re-syncing capabilities to ensure proper uninterrupted operation.

3.1 Sync Algorithm Description

The technique selected here to synchronize the receiver and the transmitter is similar to that used in the T1 Carrier System. A bit pattern called the synchronization pattern is selected. One bit at a time from the predetermined pattern is interjected at regular intervals into the bit stream generated at the transmitter. In this system, the sync bit is inserted at the beginning of each frame of 189 bits generated by processing a block of N_B samples. The receiver tries to locate the sync pattern embedded in the received bit stream, thereby locating the frame boundaries.

Any bit pattern can be used for the sync pattern as long as it does not coincide with some naturally generated pattern at the transmitter. Using a shorter sync pattern reduces the memory space and computation required at the receiver during sync acquisition. After some consideration, a two bit pattern 01 was selected for the synchronization pattern.

The following subsections detail the algorithm for the two aspects of synchronization. A short analysis is also presented with each to get some idea of the performance of these operations.

3.1.1 Synchronization Acquisition

There are two considerations in selecting this algorithm. First, it should not take too long for each acquisition operation. And secondly, the probability of making the right decision should be reasonably high to ensure that the right synchronization is achieved in a couple of attempts, if not in one.

The sync acquisition algorithm consists of segmenting the received bit stream into blocks of 189 bits each. One of the 189 bits is the sync bit,

and the corresponding bit position follows the sync pattern over the blocks of received bits. The decoder generates the sync pattern at the receiver, and checks its correlation with each of the 189 bit positions. The position that correlates exactly with the sync pattern for 10 blocks is picked to be the sync bit, marking the beginning of subsequent frames.

For sync acquisition to be unambiguously successful, only the sync bit of the 189 bits in a frame must correlate exactly with the sync pattern for 10 blocks. If there were no transmission errors, the problem here would consist of picking a deterministic sequence from the midst of a stochastic process. However, the received sync pattern is corrupted by transmission errors, an average error rate of 1%. Each of the other 188 bits are random 1's and 0's, and they correlate with the sync pattern with a probability of 0.5. With these and the assumption that the channel affects the bits independently, the probability of making a correct decision about sync acquisition can be determined.

The probability that the sync bit is transmitted without errors for 10 consecutive blocks is

$$\alpha = (0.99)^{10} = 0.9044$$

The probability that one of the other bit positions correlates perfectly with the sync pattern for 10 consecutive blocks is

$$\beta = (0.5)^{10} = 0.000976$$

The probability that none of the other 188 bit positions correlates perfectly with the sync pattern is $(1-\beta)^{188}$. The probability of an unambiguous sync acquisition decision is

$$\alpha(1-\beta)^{188} = 0.9044 \times 0.8322 = 0.7526$$

Thus, the probability of successful sync acquisition in several attempts can be computed:

In one attempt, $P\{\text{sync acquisition}\} = 0.7526$, $P\{\text{failure}\} = 0.2474$

In two attempts, $P\{\text{sync acquisition}\} = 0.9388$, $P\{\text{failure}\} = 0.0612$

In three attempts, $P\{\text{sync acquisition}\} = 0.9848$, $P\{\text{failure}\} = 0.0151$

The implementation of this algorithm is slightly different from the description here. This was done to reduce the amount of computation required.

3.1.2 Synchronization Monitor

There are three considerations in selecting the algorithm here. The sync monitor checks to see if the received sync bits follow the sync pattern. Because of transmission errors, there are some errors in the received sync bits inspite of the error control. After allowing for these errors, the algorithm should decide that sync is retained. The probability of erroneously deciding that sync is lost should be extremely small to ensure that the receiver operates uninterrupted for long periods of time. Secondly, if sync is lost, the algorithm should realize this in a reasonably short time. And finally, the algorithm should be computationally simple to implement.

With these considerations in mind, the following algorithm is suggested. A correlation between the received sync bit S_i and the expected sync bit \hat{S}_i is computed. Based on the correlation, a value is assigned to a r.v. x_i .

$$x_i = \begin{cases} +2 & \text{if } S_i \neq \hat{S}_i \\ -1 & \text{if } S_i = \hat{S}_i \end{cases}$$

This variable is used to update a sync variable v_i .

$$v_i = v_{i-1} + x_i$$

If at any time, the variable v_i exceeds the threshold T , it is decided that sync has been lost. The variable v_i starts with an initial value 0, and is constrained to be non-negative for all i . If its value drops below 0, it is reinitialized to 0. The threshold T used here is 12.

This algorithm has the effect of switching the rate of change of the sync variable from +2 when $(S_i \neq \hat{S}_i)$ to -1 when $(S_i = \hat{S}_i)$. If the channel error rate r is less than 1/3, the rate of change is negative. The sync variable decays to 0 and stays there. If the channel error rate is greater than 1/3,

the rate of change is positive and the sync variable drifts towards the threshold T . The maximum positive slope is 2 and occurs when the channel error rate is 1.

The rate of change x_i is a binary random variable. Over a period of time, its average is the slope Δv of the sync variable. The slope Δv can be used to determine the time before sync is lost.

$$t = T/\Delta v$$

The slope Δv is a binomial random variable. Its function t , the time before sync is lost, which is also a random variable can be described by a distribution similar to the binomial distribution.

$$P(t) = \frac{\Gamma(t+1)}{\Gamma(t/3+5) \Gamma(2t/3-3)} r^{(t/3+4)} (1-r)^{(2t/3-4)}, t \geq 6$$

Its expected value is

$$\langle t \rangle = T/(3r - 1)$$

Using these, some estimates of the performance of the sync monitor algorithm can be obtained. While proper sync is retained, the error rate for the sync bit is reduced to 0.005 by the error control. At this error rate, the algorithm would never lose sync. If the error rate deviated from its average value to 1, it would take 6 frames for sync to be lost. The probability of this can be computed to be 1.5625×10^{-14} .

If sync is actually lost, the average error rate for the sync bit is 0.5. The algorithm would take an average of 20 frames before deciding it has lost sync.

The implementations of these two algorithms are described in Chapter 12. The sync acquisition algorithm is implemented slightly differently from its

description here to reduce the computation involved. The sync monitor algorithm is implemented as described here using simple logical and shift operations.

CHAPTER 4

PITCH EXTRACTION STUDIES

4.1 Introduction

The aim of efficient coding methods is to reduce the channel capacity required to transmit a signal with specified fidelity. To achieve this objective, it is desirable to reduce the redundancy of the transmitted signal. One well-known procedure for removing signal redundancy is predictive coding. In predictive coding, redundancy is removed by subtracting from the signal that part which can be predicted from its past. The PARC system is essentially APC system which includes a pitch extraction loop for long-term redundancy removal.

In this chapter, several studies concerning pitch redundancy removal in PARC are described. The correlation technique, as well as AMDF algorithm, for pitch extraction is outlined in Section 4.2. The complete algorithm with pitch extraction loop was simulated on a digital computer; simulation results are discussed in Section 4.3.

4.2 Pitch Extraction Algorithms

It is well known that voiced speech is highly correlated from pitch period to pitch period [1]. The long term prediction of $s(k)$ is given by

$$\hat{s}(k|k-T) = \beta s(k-T) \quad (4.1)$$

Here β is a scalar which depends on the correlation between $s(k)$ and $s(k-T)$ while T is an estimate of the pitch period (in samples). The use of β reflects the amplitude changes of speech signal which occur from period to period especially during the beginning and end of the voiced segments. For unvoiced speech, β is generally small and long-term prediction is relatively ineffective. The long-term prediction $\hat{s}(k|k-T)$ is subtracted from $s(k)$ to form the pitch-reduced-speech $v(k) = s(k) - \hat{s}(k|k-T)$.

The goal in selecting β and T is to minimize the error

$$E_1 = \frac{1}{K} \sum_{j=1}^K [s(j) - \beta s(j-T)]^2 \quad (4.2)$$

Here block adaptation with block length of K has been assumed. The choice of K depends on various factors and will be discussed in a next section. The derivative of E_1 with respect to β yields

$$\frac{\partial E_1}{\partial \beta} = \frac{2}{K} \sum_{j=1}^K [s(j) - \beta s(j-T)] s(j-T) \quad (4.3)$$

Equating this derivative to zero and solving for β gives

$$\beta = \frac{\sum_{j=1}^K s(j)s(j-T)}{\sum_{j=1}^K s^2(j-T)} \quad (4.4)$$

If this result is substituted in Eq. (4.2), the equation becomes function of T alone given by

$$E_1(T) = \frac{1}{K} \sum_{j=1}^K s^2(j) - \frac{\frac{1}{K} \sum_{j=1}^K s(j)s(j-T)}{\sum_{j=1}^K s^2(j-T)} \quad (4.5)$$

Therefore to minimize E_1 with respect to T it is necessary to maximize the rightmost term of equation (4.5). The approach used was to compute

$$A(T) = \frac{\sum_{j=1}^K s(j)s(j-T)}{\sum_{j=1}^K s^2(j-T)} \quad (4.6)$$

for all values of T , $T_{\min} \leq T \leq T_{\max}$ and then select the value for which $A(T)$ is maximum. The lower limit, T_{\min} , of the search range was selected to be smaller than minimum value of pitch periods for different speakers while the upper limit, T_{\max} , is influenced by various factors such as number of bits available for transmission of T 's, processing time limitations due to real time application and maximizing energy reduction.

The above method, though simple to implement, involves extensive computation. For example, if the block length is K and searching range is R then for each value of R there are $2K+2$ multiplications and K additions. Hence the total number of multiplications for finding the pitch period becomes $R(2K+2)$; if $R=100$ and $K=100$ then this number is 20200 (this is just for one block of 100 samples. This many multiplications consume significant processing time which is crucial in real time implementation in Macro Arithmetic Processor (MAP).

The stringent requirement on timing in the MAP, led to a modification of the above correlation technique for pitch extraction. This is done by forming Average Magnitude Difference Function (AMDF)[2]

$$A'(T) = \sum_{j=1}^K |s(j) - s(j-T)| \quad (4.7)$$

$$T_{\min} \leq T \leq T_{\max}$$

It is easy to see that for any periodic function, the above sum is minimum for T equal to the period. Hence, the pitch parameter T was determined by minimizing the function $A'(T)$ with respect to T . The gain parameter was obtained by substituting this value of T into Eq. (4.4). The computational saving in this method is apparent since there is no multiplication involved.

This modification of correlation method gives exactly the same values of T (and hence of β) in voiced speech but differs in unvoiced speech. However, unvoiced speech is non-periodic and T is arbitrary and hence not important.

Figure 4.1 shows the plot of β and T against speech samples. The correlation coefficient β jumps to a high value for a voiced speech block which is followed by silence. This high value amplifies the quantization noise thus decreasing overall signal to noise ratio. Limiting β to $[-2, 2]$ was found to eliminate this problem and give satisfactory performance. Further discussion of limiting in context with quantization of β and coding is presented in Chapter 11.

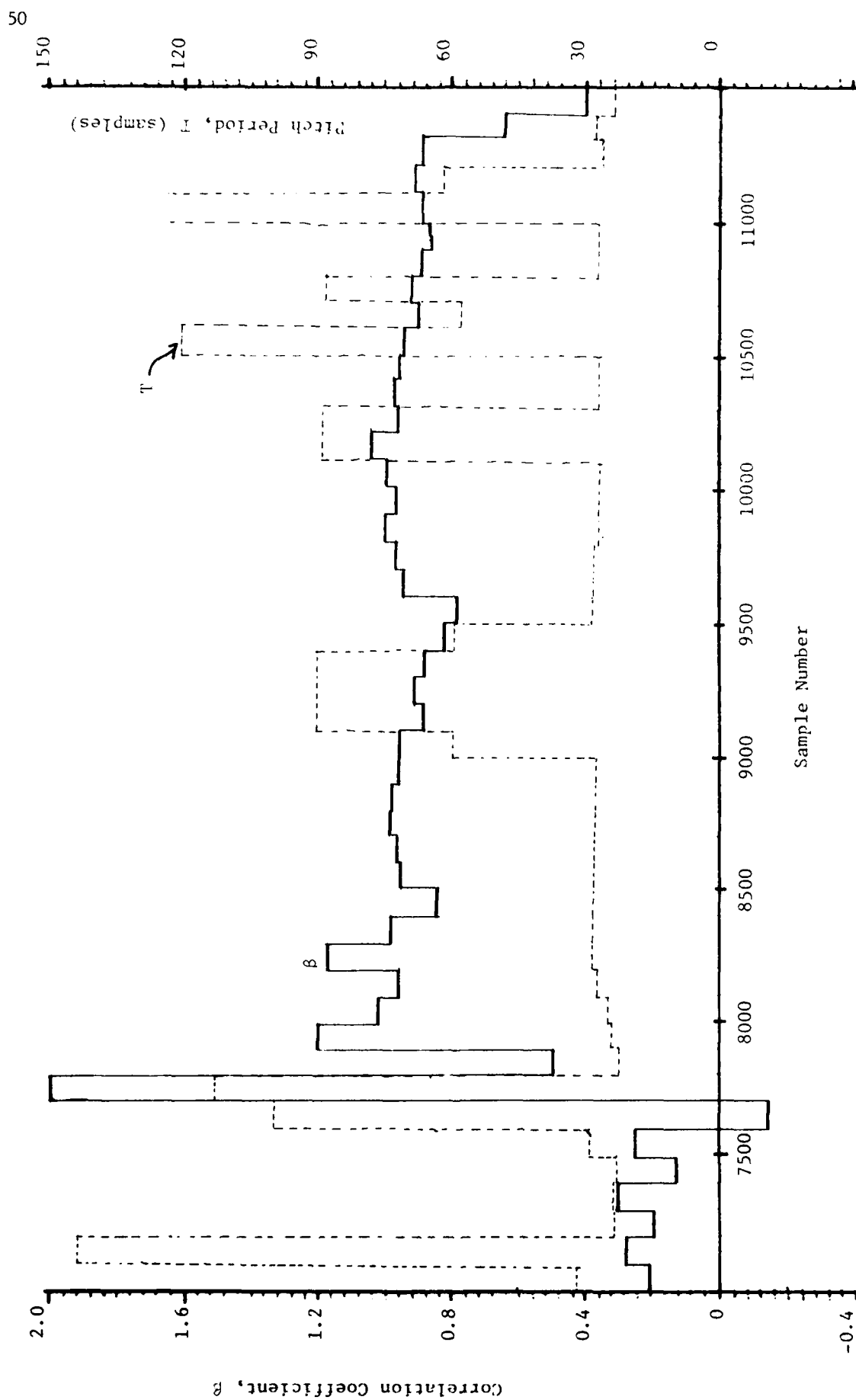


Fig. 4.1 Pitch Period T and Correlation Coefficient β vs Sample Number. (Sent 1: Male Speaker)

PARC algorithm was simulated on PDP 11/60 computer. The following phonetically balanced sentences were chosen for the simulation work.

Male speaker: sent 1 - "Cats and Dogs each hate the other."

Female speaker: sent 11 - "The pipe began to rust while new."

Beta's and T's were extracted using both the correlation and AMDF methods discussed above. The results of this study are shown in Table 4.1.

TABLE 4.1a

Comparison of Correlation and AMDF Method
for Pitch Extraction

Sentence 11: Female Speaker

| Sample No. | Correlation Technique | | AMDF Algorithm | |
|------------|-----------------------|----|----------------|-----|
| | β | T | β | T |
| 3500 | -0.26 | 23 | 0.06 | 50 |
| 3600 | -0.23 | 25 | 2.00 | 134 |
| 3700 | 0.42 | 22 | 0.57 | 147 |
| 3800 | 0.85 | 58 | 0.85 | 58 |
| 3900 | 1.03 | 29 | 1.03 | 29 |
| 4000 | 0.95 | 29 | 0.95 | 29 |
| 4100 | 0.91 | 88 | 0.91 | 88 |
| 4200 | 0.93 | 88 | 0.93 | 88 |
| 4300 | 0.99 | 29 | 0.99 | 29 |
| 4400 | 0.99 | 29 | 0.99 | 29 |
| 4500 | 0.80 | 29 | 0.80 | 29 |
| 4600 | 1.02 | 29 | 1.02 | 29 |
| 4700 | 1.00 | 29 | 1.00 | 29 |
| 4800 | 0.92 | 29 | 0.92 | 29 |
| 4900 | 0.61 | 28 | 0.61 | 28 |
| 5000 | 0.31 | 25 | 0.24 | 89 |
| 5100 | 0.32 | 20 | 0.32 | 20 |
| 5200 | 0.56 | 94 | 0.41 | 120 |
| 5300 | 0.30 | 62 | 0.24 | 23 |
| 5400 | 0.93 | 27 | 0.93 | 27 |
| 5500 | 0.87 | 29 | 0.87 | 29 |
| 5600 | 0.89 | 30 | 0.89 | 30 |
| 5700 | 0.99 | 30 | 0.99 | 30 |
| 5800 | 1.01 | 30 | 1.01 | 30 |
| 5900 | 1.01 | 30 | 1.01 | 30 |
| 6000 | 1.07 | 30 | 1.07 | 30 |
| 6100 | 1.07 | 30 | 1.07 | 30 |
| 6200 | 1.08 | 30 | 1.08 | 30 |
| 6300 | 1.04 | 30 | 1.04 | 30 |
| 6400 | 1.01 | 30 | 1.01 | 30 |
| 6500 | 0.97 | 30 | 0.97 | 30 |
| 6600 | 0.94 | 30 | 0.94 | 30 |
| 6700 | 0.86 | 30 | 0.86 | 30 |
| 6800 | 0.79 | 30 | 0.79 | 30 |
| 6900 | 0.66 | 31 | 0.66 | 31 |
| 7000 | 0.19 | 63 | 0.04 | 27 |

TABLE 4.1b

Comparison of Correlation and AMDF Method
for Pitch Extraction

Sentence 1: Male Speaker

| Sample No. | Correlation Technique | | Sample No. | AMDF Algorithm | |
|------------|-----------------------|-----|------------|----------------|-----|
| | β | T | | β | T |
| 1700 | 0.53 | 141 | 1700 | 0.37 | 38 |
| 1800 | 0.35 | 141 | 1800 | 0.35 | 141 |
| 1900 | 0.48 | 135 | 1900 | 0.32 | 22 |
| 2000 | 0.11 | 148 | 2000 | -0.25 | 133 |
| 2100 | 0.30 | 27 | 2100 | 0.20 | 145 |
| 2200 | 1.45 | 42 | 2200 | 1.45 | 42 |
| 2300 | 0.71 | 48 | 2300 | 0.71 | 48 |
| 2400 | 0.76 | 50 | 2400 | 0.76 | 50 |
| 2500 | 0.93 | 52 | 2500 | 0.93 | 52 |
| 2600 | 0.74 | 53 | 2600 | 0.74 | 53 |
| 2700 | 0.96 | 53 | 2700 | 0.96 | 53 |
| 2800 | 0.94 | 53 | 2800 | 0.94 | 53 |
| 2900 | 0.94 | 54 | 2900 | 0.94 | 54 |
| 3000 | 0.94 | 55 | 3000 | 0.94 | 55 |
| 3100 | 0.90 | 57 | 3100 | 0.90 | 57 |
| 3200 | 0.72 | 59 | 3200 | 0.72 | 59 |
| 3300 | 0.60 | 61 | 3300 | 0.60 | 61 |
| 3400 | 0.54 | 62 | 3400 | 0.52 | 63 |
| 3500 | 0.79 | 24 | 3500 | -0.98 | 31 |
| 3600 | 1.16 | 52 | 3600 | 1.16 | 52 |
| 3700 | 1.21 | 53 | 3700 | 1.21 | 53 |
| 3800 | 1.01 | 53 | 3800 | 1.01 | 53 |
| 3900 | 0.89 | 53 | 3900 | 0.89 | 53 |
| 4000 | 0.81 | 55 | 4000 | 0.81 | 55 |
| 4100 | 0.97 | 57 | 4100 | 0.97 | 57 |
| 4200 | 0.98 | 58 | 4200 | 0.98 | 58 |
| 4300 | 1.00 | 58 | 4300 | 1.00 | 58 |
| 4400 | 0.96 | 59 | 4400 | 0.96 | 59 |
| 4500 | 0.85 | 59 | 4500 | 0.86 | 60 |
| 4600 | 0.92 | 60 | 4600 | 0.92 | 60 |
| 4700 | 0.84 | 61 | 4700 | 0.84 | 61 |
| 4800 | 0.74 | 62 | 4800 | 0.75 | 63 |
| 4900 | 0.69 | 65 | 4900 | 0.69 | 65 |
| 5000 | 0.60 | 66 | 5000 | 0.60 | 66 |
| 5100 | 0.60 | 68 | 5100 | 0.61 | 67 |
| 5200 | 0.42 | 65 | 5200 | 0.44 | 67 |
| 5300 | 0.93 | 64 | 5300 | 0.93 | 64 |
| 5400 | 0.71 | 64 | 5400 | 0.71 | 64 |
| 5500 | 0.87 | 65 | 5500 | 0.88 | 66 |
| 5600 | 0.66 | 131 | 5600 | 0.66 | 131 |
| 5700 | 0.33 | 66 | 5700 | 0.34 | 67 |
| 5800 | 0.32 | 62 | 5800 | 0.32 | 62 |

4.3 Redundancy Removal

As described earlier, the APC system is based on the removal of two kinds of redundancy: short term redundancy caused by vocal tract filter and long-term redundancy caused by pitch frequency. Once the pitch period T and gain parameter β are determined, reduced speech is formed as

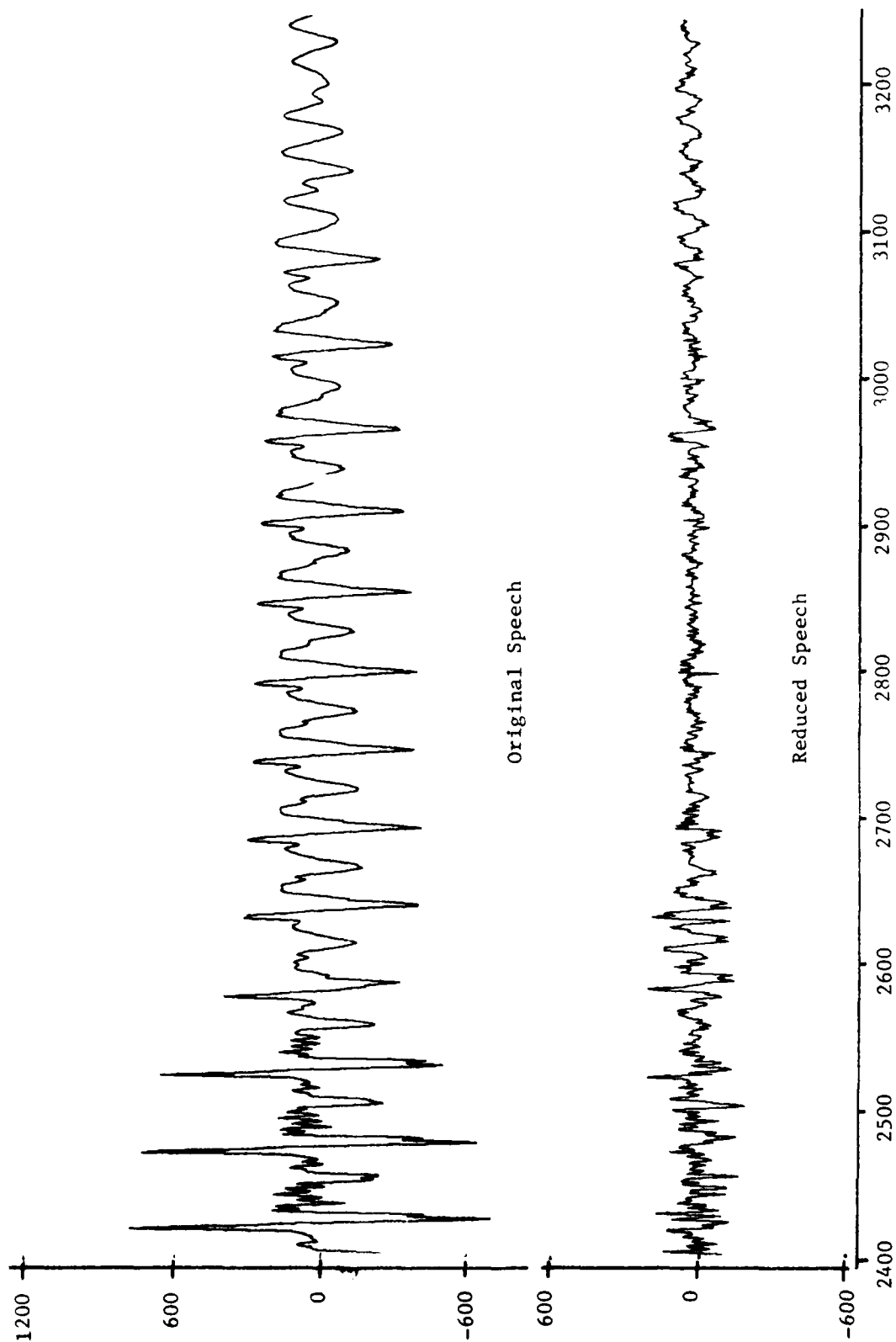
$$v(k) = s(k) - \beta \hat{s}(k|k-T) \quad (4.8)$$

where $\hat{s}(k|k-T)$ represents the reconstructed speech sample. Figure 4.2 shows the plot of reduced and original speech. It is easy to notice the energy reduction achieved in the almost periodic voiced portion of the speech. The amount of energy reduction achieved is expressed by SER (Signal Energy Reduction) which is calculated as

$$SER = -10 \log_{10} \frac{\sum v^2(k)}{\sum s^2(k)} \quad (4.9)$$

As the value of signal energy reduction is increased the dynamic range of the input signal to quantizer is reduced; hence the reduced speech signal can be represented by the lower quantizer levels thus requiring fewer bits for transmission. The SER can be increased by accurately picking pitch period T and choosing the block size such that effect of transition of β from block to block is minimum. The parameters β and T are associated with every block. For smaller block sizes, the number of parameters to be transmitted per second is increased. However, for smaller block size, the amplitude variations are closely represented by β and transition of β from block to block is smooth. These factors contribute to improve SER.

Table 4.2 shows the effect of block-size variation on SER. The following performance measures were also computed:



Sample Number
Fig. 4.2.1 Sentence 1, Male Speaker

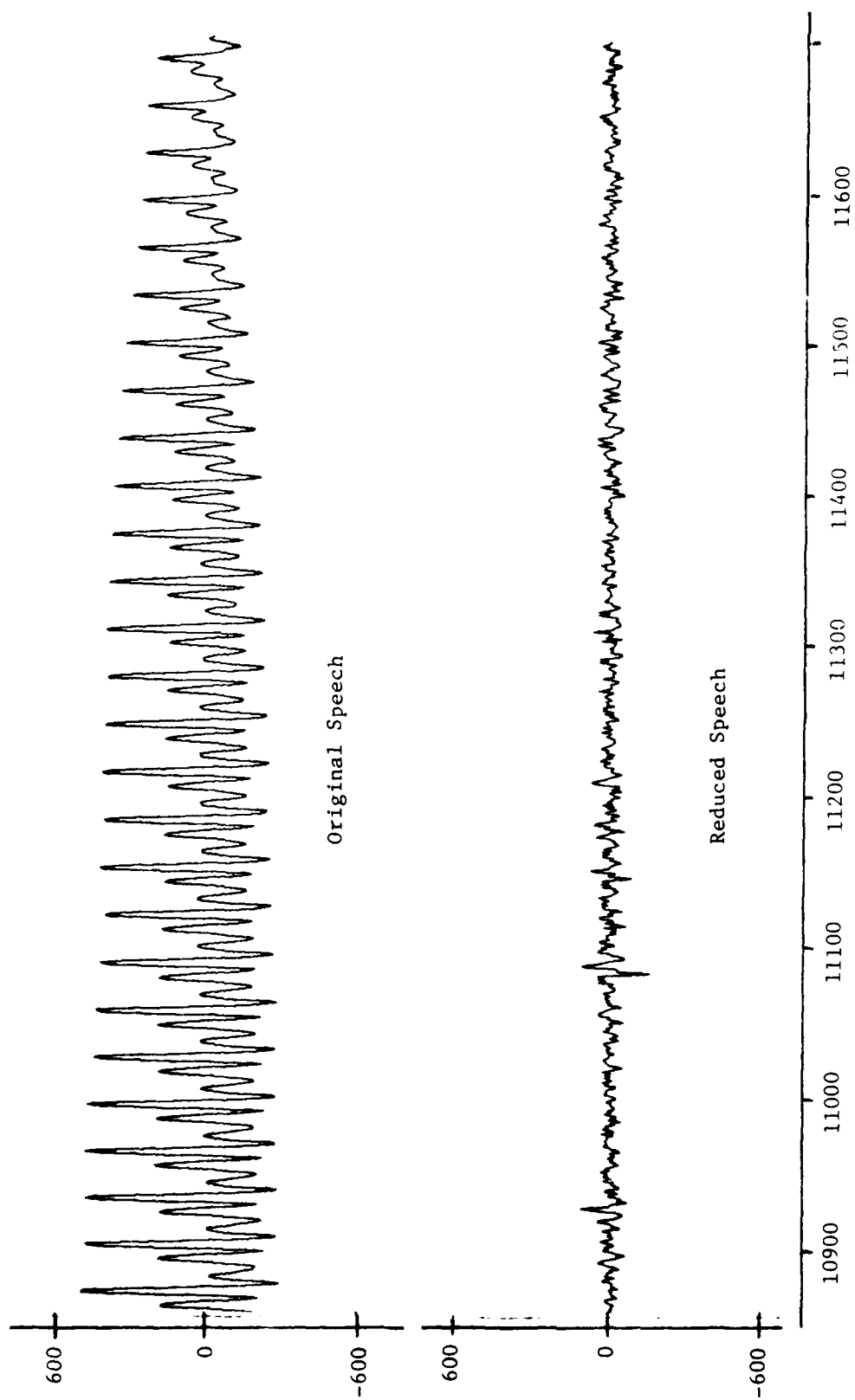


Fig. 4.2b Sentence 11, Female Speaker

Table 4.2

Effect of Block Size on Various Performance Measures

| Sentence | Block Size | SER | Overall SNR | Inloop SNR | Entropy bits/sample |
|----------|------------|----------|-------------|------------|---------------------|
| 1 | 20 | 8.47 db | 20.81 db | 12.34 db | 1.53 |
| | 40 | 7.07 db | 20.24 db | 13.17 db | 1.48 |
| | 80 | 5.63 db | 19.08 db | 13.45 db | 1.46 |
| | 100 | 5.44 db | 18.97 db | 13.53 db | 1.43 |
| | 120 | 5.15 db | 18.85 db | 13.70 db | 1.43 |
| | 140 | 4.70 db | 18.26 db | 13.56 db | 1.44 |
| | 160 | 4.56 db | 18.33 db | 13.77 db | 1.45 |
| | 180 | 4.41 db | 18.31 db | 13.90 db | 1.42 |
| | 200 | 4.07 db | 18.20 db | 14.13 db | 1.43 |
| 11 | 20 | 11.19 db | 22.78 db | 11.59 db | 1.48 |
| | 40 | 9.81 db | 22.44 db | 12.63 db | 1.44 |
| | 100 | 8.48 db | 21.72 db | 13.24 db | 1.43 |
| | 120 | 8.04 db | 21.65 db | 13.61 db | 1.42 |
| | 140 | 8.20 db | 21.57 db | 13.37 db | 1.42 |
| | 160 | 7.92 db | 21.59 db | 13.67 db | 1.43 |
| | 180 | 8.16 db | 21.64 db | 13.48 db | 1.43 |
| | 200 | 8.33 db | 21.58 db | 13.25 db | 1.44 |

$$\text{Overall SNR} = 10 \log_{10} \frac{\sum s^2(k)}{\sum [s(k) - \hat{s}(k)]^2} \quad (4.10)$$

$$\text{Inloop SNR} = 10 \log_{10} \frac{\sum v^2(k)}{\sum [v(k) - \hat{v}(k)]^2} \quad (4.11)$$

where $v(k) = \hat{v}(k) + n_q(k)$

$$\text{Entropy } H = - \sum_{i=1}^{11} p_i \log_2 p_i \quad (4.12)$$

where p_i = Probability of occurrence of i^{th} quantizer level.

As SER improves overall performance also improves. It is interesting to note that the SNR (overall) may increase even if the SNR (inloop) decreases because of the improvement in SER. The speech signal spectrum becomes flatter because of pitch extraction thus adversely affecting the performance of predictor in INLOOP. However, because of smaller dynamic range of reduced speech, the quantizer noise is decreased which more than compensates for the poor performance of predictor. Hence the overall performance improves.

The searching range ($T_{\max} - T_{\min}$) for pitch extraction also affects the redundancy removal. It was observed that a longer searching range gives better SER while a small value of searching range decreases SER by as much as 2 db. A searching range of the order of twice the maximum pitch period appears to be sufficient. However, the longer searching range also means more computations and hence more CPU time. In the real-time simulations on the MAP, timing is critical and therefore the number of computations and memory transfers need to be reduced. In such cases the searching range must be reduced to achieve a compromise between the number of computations

and the reduction in SER that can be tolerated.

It was noticed that the pitch extraction algorithm sometimes picks double or triple pitch periods. This fact has only a modest effect redundancy removal. However, the transmission of double or triple pitch period values may require the allocation of more bits for transmission of T . Again, it is desirable to limit the search range.

In Section 4.2, the estimation algorithm used to compute the pitch period T and long term gain is based solely on the original speech. In fact, as seen by examining Fig. 2.10 the long-term redundancy removal operation actually subtracts the reconstructed speech from original speech. In an attempt to compensate for this fact, the β obtained from Eq. (4.4) was modified by multiplying by scalar α as

$$\beta^* = \alpha \beta \quad (4.13)$$

Here α can be expressed [3] as

$$\alpha = \frac{1}{1 + \alpha} \quad (4.14)$$

where α is the inverse of signal to noise ratio. The parameter α was varied between 0 and 1.2 with no significant improvement was noticed. See Table 4.3.

TABLE 4.3

The Effect of α on SER and Overall SNR

| α | SER | Overall SNR |
|----------|---------|----------------|
| 0 | 0 db | 15.61 db |
| 0.9 | 5.45 db | 19.23 db |
| 0.94 | 5.55 db | 19.29 db |
| 0.96 | 5.57 db | 19.24 db |
| 0.98 | 5.55 db | 19.40 db |
| 0.995 | 5.52 db | 19.28 db |
| 1.0 | 5.55 db | 19.31 db |
| 1.04 | 5.48 db | 19.16 db |
| 1.1 | 5.37 db | 19.21 db |
| 1.2 | 5.07 db | 18.79 db |

4.4 References

- [1] Flanagan, J. L.: Speech Analysis, Synthesis and Perception, 2nd Ed., Springer-Verlag, N.Y., 1972.
- [2] Ross, M., et al.: "Average Magnitude Difference Function Pitch Extractor," IEEE Trans. on Acoustics, Speech and Signal Processing, Vol. ASSP-22, pp 353-362, Oct. 1974.
- [3] "Study of Sequential Estimation Methods for Speech Digitization", Department of Electrical Engineering, University of Notre Dame, Notre Dame, Indiana, June 1975.

CHAPTER 5

TREE CODING

5.1 Introduction

The concept known as tree coding was investigated as part of this study to evaluate its ability to improve performance. One particular algorithm, the (M,L) algorithm, was the basis for most of the investigation. A modified version of the (M,L) algorithm called adaptive tree searching was developed and investigated. Simulations indicated that adaptive tree searching marginally improved the performance of the PARC algorithm.

5.2 The (M,L) Algorithm

5.2.1 Description

The (M,L) algorithm is one of a number of algorithms which perform what is known as tree coding. The main idea of tree coding is to defer making a decision, in this case, which quantizer level should be used for a given sample, until a later time when it can be made in light of that which follows. Tree coding is useful in predictive or backward adaptive quantization systems, because the selection of a quantizer level affects the selection of quantizer levels in the future. This effect can be represented graphically in the form of a tree, where a node represents the "state" of the system as a result of selecting the sequence of quantizer levels leading to that "state", with a branch connecting the node to the node representing the previous "state". The tree is rooted by an arbitrary "state" at an arbitrary time, and evolves in time, with a new level of nodes added at each sample time.

The (M,L) algorithm operates in the following way, and is illustrated in Figure 5.1. At a given time, let us say that there are n nodes in the outermost level of the tree, and a new sample is to be quantized. A new level of nodes are then "grown" from the outermost level, representing all of the new possible "states". Thus, if there are k quantizer levels, there will be kn nodes in the new level. These new nodes are then ranked by some performance criterion, such as quantization noise. Nodes are next pruned from the tree. This is a key step, because it is this that prevents the task from growing exponentially. In order to insure that a quantizer level is selected for a sample in a finite amount of time, a

quantizer level decision is forced, if necessary, for the sample L time units ago, by picking the predecessor node, in the level L -levels in, of the best of the new nodes. All branches which do not stem from that predecessor are pruned. If more than M new nodes remain, the M best nodes are kept, and the rest are pruned. The process then starts over.

5.2.2 Results

The (M,L) algorithm has been studied by a number of investigators in connection with simpler quantization schemes, such as adaptive delta modulation (ADM) and adaptive differential pulse code modulation (ADPCM). For example, Jayant and Christensen reported a 3dB improvement in the signal-to-noise ratio (SNR) using the (M,L) algorithm with simple ADM and ADPCM schemes, with $M=4$ and $L=7$. In contrast, our investigation concentrated on the use of the (M,L) algorithm in connection with adaptive residual coding (ARC). It was originally envisioned that the resulting algorithm would be embedded within a pitch extraction loop. Some preliminary work was done, however, without the pitch extraction loop in order to verify the tree coding software and to gain some insight into its operation. For example, by using a four level fixed quantizer and a third order fixed predictor in the ARC algorithm, an improvement of 3dB in the SNR was achieved for $M=4$ and $L=7$. Unfortunately, the results obtained using the standard ARC algorithm did not show as great an improvement in SNR.

In order to make the following results more understandable though, a comment is necessary here. One of the key elements of the ARC algorithm is the source coding, which translates quantizer levels into bit patterns. The selection of a good source code is dependent, however, on the statistics of that which is to be encoded. As a result, the performance of the (M,L) algorithm was evaluated without use of a source coder. Instead, the entropy of the quantizer levels was computed, which allows for a fair comparison between the possibilities. In general, a higher entropy allows better performance.

Some of the first results obtained were for the first second of Sentence

1, "Cats and dogs each hate each other". In this series of runs, a five level quantizer was used in the standard ARC algorithm, M was set to five, and L was varied. The results are shown in Table 5.1. These results illustrate the typical effect of tree coding: increasing L improves performance by increasing the SNR and decreasing the entropy simultaneously. The results also illustrate another phenomenon of tree coding: that tree coding becomes less effective in improving performance as the performance of the base algorithm improves.

More results were obtained for the first second of Sentence 1 using a 19 level quantizer. The results are shown in Table 5.2. These results again show that, in general, increasing L or M (or both) improves performance, but that the amount of improvement is smaller than that achievable by the poorer-performing five level quantizer. There are, however, several instances where it appears that increasing L or M has decreased performance. This can occur because tree coding can make a suboptimal decision because the set of possible decisions is deliberately limited. As a result, tree coding can occasionally get "fooled".

Simulations were also performed with the tree coding ARC algorithm embedded within a pitch extraction loop. Some typical results are shown in Table 5.3. These results were obtained from Sentence 1, using an 11 level quantizer and $M=5$. Performance is again generally improved by tree coding, but by only a small amount.

Table 5.1. Results for a five level quantizer, $M=5$.

| <u>L</u> | <u>SNR (dB)</u> | <u>ENTROPY (BITS/SAMPLE)</u> |
|----------|-----------------|------------------------------|
| 1 | 14.03 | 1.423 |
| 2 | 14.60 | 1.421 |
| 4 | 14.66 | 1.407 |
| 8 | 14.98 | 1.403 |
| 16 | 15.08 | 1.393 |

Table 5.2. Results for a 19 level quantizer.

| <u>M \ L</u> | <u>1</u> | <u>2</u> | <u>3</u> | <u>4</u> | <u>5</u> | <u>6</u> | <u>7</u> | <u>8</u> | <u>9</u> | <u>10</u> |
|--------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|
| 1 | 21.30 2.263 | 21.30 2.262 | 21.30 2.262 | | | | | | | |
| 2 | 21.30 2.263 | 21.45 2.258 | 21.57 2.250 | 21.69 2.250 | 21.61 2.265 | 21.88 2.244 | 21.89 2.242 | 21.89 2.248 | 21.71 2.245 | 21.80 2.246 |
| 3 | 21.30 2.263 | 21.45 2.258 | 21.57 2.250 | 21.72 2.246 | 21.90 2.249 | | | | | |
| 4 | | 21.45 2.258 | 21.63 2.248 | 21.63 2.243 | 21.89 2.244 | | | | | |
| 5 | | 21.44 2.257 | 21.64 2.253 | 21.75 2.244 | 21.75 2.243 | | | | | |

Key: upper number is SNR (dB), lower number
is entropy (bits/sample)

Table 5.3. Results for an 11 level quantizer, with
M=5, embedded in a pitch removal loop.

| <u>L</u> | SNR <u>inside pitch loop (dB)</u> | SNR <u>overall (dB)</u> | Entropy <u>(bits/sample)</u> |
|----------|--------------------------------------|----------------------------|---------------------------------|
| 1 | 13.75 | 18.41 | 1.372 |
| 2 | 14.01 | 18.63 | 1.355 |
| 3 | 14.29 | 18.89 | 1.369 |
| 4 | 14.36 | 18.97 | 1.353 |
| 5 | 14.22 | 18.83 | 1.350 |
| 6 | 14.42 | 19.05 | 1.353 |

5.3 Adaptive Tree Coding

5.3.1 Description

Adaptive tree coding was developed as an attempt to gain the improvement in performance of the (M,L) algorithm, without using as many computations. It was felt that the performance improvement was desirable, but the large number of computations was a costly trade-off, and, more importantly, unable to be done in real time. This led to the search for some way of improving the performance/computation ratio.

Adaptive tree coding was the result of that search. It is based on the fact that the receiver does not need to know to what extent, if any, that tree coding is being performed at the transmitter. The receiver acts only upon the quantizer levels it receives; it does not matter how those quantizer levels were arrived at. So the basic concept of adaptive tree coding is to use tree coding only when it appears to make sense.

Two strategies for adaptive tree coding were developed from this basic concept. The first strategy was to perform additional tree pruning, so that a node would have to have a reasonable chance of being selected to be kept. Specifically, a node would be pruned if the value of the criterion for it were worse than the value of the criterion for the best node, multiplied by an arbitrary factor. In this way, when growing the next level of nodes, time would not be spent growing nodes which had a high probability of being pruned eventually.

The second strategy was to "turn off" the coding (by setting M equal to 1) when the system was performing well, and "turn it back on" when it was not performing well. The idea behind this strategy was that if the system were

performing well, there was little to be gained from tree coding. This strategy was implemented by checking the value of the criterion for the best new node was better than some arbitrary threshold. If, on the other hand, tree coding was not in use, then tree coding would not be used until the value of the criterion for the best new node was worse than some second arbitrary threshold.

Both of these strategies were employed in our simulation of adaptive tree coding, because they are somewhat complementary in nature: the second strategy provides "course tuning", and the first strategy provides "fine tuning".

5.3.2 Results

Results were first obtained for adaptive tree coding using only the second ("threshold") strategy. Some typical results from these runs are shown in Table 5.4. Tree coding was used for 26% of the samples in Sentence 1, and for 14% of the samples in Sentence 11. The results indicate the ability of adaptive tree coding to improve performance with a small increase in computation.

Results were next obtained for adaptive tree coding using only the first ("factor") strategy. A sample of the results are shown in Table 5.5. On these runs, Sentence 1 was used as the input, and M was set to 5. A measure of the decrease in computations is "effective M", which is the average number of nodes retained for each new sample. It can clearly be seen that the number of calculations can be decreased dramatically while still retaining much of the increased performance. In fact, it can be seen from the data that the performance was improved by adaptive tree coding. One possible explanation for this phenomenon would be that for those cases where adaptive tree coding increased performance, that the additional pruning eliminated nodes which appeared to be good, but were not in the long run.

Finally, a series of simulation runs were made which utilized both adaptive strategies simultaneously. The results indicated that the two strategies were complementary, in that use of both was better than the use of either alone. Some representative results are shown in Table 5.6. It can be seen that about the same performance is obtained by adaptive tree coding as with tree coding, with about a 66% reduction in computations.

Table 5.4 Results using the "threshold" strategy for adaptive tree coding.

Sentence 1 (male speaker):

| | <u>PARC</u> | <u>Adaptive tree coding</u> | <u>Tree coding</u> |
|----------------------------|-------------|---------------------------------|--------------------|
| SNR inside pitch loop (dB) | 13.77 | 14.13 | 14.42 |
| SNR overall (dB) | 18.42 | 18.75 | 19.04 |
| Entropy (bits/sample) | 1.374 | 1.374 | 1.356 |

Sentence 11 (female speaker):

| | <u>PARC</u> | <u>Adaptive tree coding</u> | <u>Tree coding</u> |
|----------------------------|-------------|---------------------------------|--------------------|
| SNR inside pitch loop (dB) | 12.84 | 13.35 | 13.64 |
| SNR overall (dB) | 20.88 | 21.40 | 21.70 |
| Entropy (bits/sample) | 1.386 | 1.374 | 1.338 |

Table 5.5 Results using the "factor" strategy for adaptive tree coding.

| <u>FACTOR</u> | <u>Overall SNR(dB)</u> | <u>Entropy (bits/sample)</u> | <u>Effective M</u> |
|---------------|----------------------------|----------------------------------|------------------------|
| 1.E6 (60dB) | 18.92 | 1.347 | 5.00 |
| 10.00(10dB) | 18.92 | 1.347 | 4.46 |
| 7.94(9dB) | 18.92 | 1.348 | 4.43 |
| 6.31(8dB) | 18.92 | 1.348 | 4.39 |
| 5.01(7dB) | 18.92 | 1.348 | 4.33 |
| 3.98(6dB) | 18.92 | 1.348 | 4.24 |
| 3.16(5dB) | 18.94 | 1.349 | 4.10 |
| 2.51(4dB) | 19.09 | 1.343 | 3.84 |
| 2.00(3dB) | 19.08 | 1.348 | 3.41 |
| 1.58(2dB) | 19.07 | 1.356 | 2.62 |
| 1.50(1.75dB) | 19.05 | 1.354 | 2.40 |
| 1.41(1.50dB) | 18.94 | 1.360 | 2.10 |
| 1.33(1.25dB) | 18.94 | 1.347 | 1.85 |
| 1.26(1dB) | 18.82 | 1.354 | 1.58 |
| 1.00(0dB) | 18.42 | 1.374 | 1.00 |

Table 5.6 Results using both strategies for adaptive tree searching.

| | <u>Overall SNR(dB)</u> | <u>Entropy (bits/sample)</u> | <u>Effective M</u> |
|---|----------------------------|----------------------------------|------------------------|
| PARC | 18.42 | 1.374 | 1.00 |
| Adaptive tree coding HITHR = 2416 (15dB), LOTHR = 304.1 (24dB), FACTOR = 1.50 (1.75dB) | 18.92 | 1.360 | 1.51 |
| Tree coding | 18.92 | 1.347 | 5.00 |

5.4 Conclusions and suggestions for further research

It would appear that on the basis of the results obtained, tree coding is an effective way to increase the performance of a quantization system. The major question though, is not whether it is effective, but how effective it is relative to what it costs. It would appear that tree coding may not be effective in this sense, but that some form of adaptive tree coding may be. The question was moot for this project, because there was not sufficient real time to perform a tree coding version of PARC.

The reason why tree coding appears to be relatively ineffective with PARC may be that the ARC algorithm has been overly adapted for PARC — that is a set of parameters optimized for PARC will most probably be a poor choice to be used with tree coding. This makes a lot of sense if you think about how tree coding can help a system. For the PARC algorithm to work well, the parameters are chosen so that typically only one quantizer level results in a reasonable amount of error — to do otherwise would be suboptimal for PARC, since only one quantizer level can be selected, and having more than one reasonable quantizer level would simply reduce the dynamic range of the quantizer or increase the average quantization error. In contrast, for tree coding to be effective, several quantizer levels should be reasonable for each sample.

In order to modify the ARC parameters to make tree coding more effective then, it would appear that it would be desirable to select parameters which would decrease granular noise and let the tree coding reduce slope overload noise. Specifically, it would seem to be desirable to "move in" the output and scaling factors of the quantizer, decrease the updating gains, and reduce the "time constants" of the system. In this way, there would be a

richer selection of quantizer levels for the tree code.

There are also several other areas to be researched. In our simulations, the tree coding took place entirely within the pitch loop. It might be interesting, therefore, to investigate the effect of basing the pruning criterion on the reconstructed speech rather than the reconstructed depitched speech. Doing this might aid in the smooth transition from one pitch block to the next. Another idea which could be investigated would be delayed updating of the ARC algorithm. In delayed updating, the algorithm would be updated on the basis of the quantizer level just decided upon (corresponding to the sample L samples earlier), so that the updating could be done once for all nodes, eliminating many computations.

Tree coding might also be made more effective by making changes in the tree coding algorithm itself. One possible path for investigation would be in the area of variable symbol release, as developed by Goris. Another possible path would involve investigating different forms of the pruning criterion. For example, it might improve performance to weigh the contribution from earlier samples more heavily than that from recent samples, because it would appear that the "soft" decisions become "harder" as time passes.

5.5 References

1. J. B. Anderson and J. B. Bodie, "Tree Encoding of Speech", IEEE Trans. on Information Theory, Vol. IT-21, No. 4, July 1975, pp. 379-387.
2. J. B. Anderson and C.-W. P. Ho, "Architecture and Construction of a Hardware Sequential Encoder for Speech", IEEE Trans. on Communications, Vol. COM-25, No. 7, July 1977, pp. 703-707.
3. J. D. Gibson and A. C. Goris, "Incremental and Variable-Length Tree Coding of Speech", Proceedings 1979 International Conference on Communications, Boston, MA, June 1979.
4. A. D. Goris, "Tree Coding of Speech Using Variable Length Symbol Release Techniques", Master's Thesis, Texas A&M University, August 1979.
5. N. S. Jayant and S. A. Christensen, "Tree-Encoding of Speech Using the (M,L)-Algorithm and Adaptive Quantization", IEEE Trans. on Communications, Vol. COM-26, No. 9, September 1978, pp. 1376-1379.
6. J. Uddenfeldt and L. H. Zetterberg, "Algorithms for Delayed Encoding in Delta Modulation with Speech-Like Signals", IEEE Trans. on Communications, Vol. COM-24, No. 6, June 1976, pp. 650-655.
7. S. G. Wilson and S. Husain, "Adaptive Tree Encoding of Speech at 8000 Bits/s with a Frequency-Weighted Error Criterion", IEEE Trans. on Communications, Vol. COM-27, No. 1, January 1979, pp. 165-170.
8. L. H. Zetterberg and J. Uddenfeldt, "Adaptive Delta Modulation with Delayed Decision", IEEE Trans. on Communications, Vol. COM-22, No. 9, September 1974, pp. 1195-1198.

CHAPTER 6

BACKWARD PITCH EXTRACTION ADAPTIVE RESIDUAL CODER (BPARC)

6.1 Introduction

Many of the practical systems for digitizing speech are variants of differential pulse code modulation (DPCM). The speech coder developed for the 9.6 Kbs bit rate uses this structure augmented by a pitch extraction loop. This algorithm is called Pitch extraction Adaptive Residual Coder (PARC).

The system described in the chapter is identical to PARC except for the method of pitch extraction. In PARC, pitch is extracted block by block from raw speech. Once the correlation coefficient β and pitch period T are known, reduced speech is formed, processed by the coder and transmitted. The β 's and T 's also need to be transmitted; the number of bits required to transmit these parameters depends on the number of parameters available (this depends on block length), type of coding employed and the way they are transmitted. In addition to the number of bits required for transmission, the transmission of these parameters necessitates a framing of the bit stream and an associated frame synchronization problem.

In order to avoid this framing problem, a backward adaptive approach was investigated which would not require transmission of β 's and T 's. Since the values of β and T change very little in a given voiced region, these parameters can be calculated for a previous block of speech and used for the current block to reconstruct speech. As noted in an earlier chapter, short pitch blocks provide the best performance. However, these short blocks require the transmission of a large amount of side information,

β and T's, making them impractical. The use of the backward identification of β and T eliminates this problem.

In following sections, this approach referred to as Backward PARC (BPARC) is described along with various computer simulation studies comparing it with the standard PARC algorithm. A complete listing of the source program and a flow chart for the simulation are also included in this chapter.

6.2 System Structure

Figure 6.1 shows the block diagram of the backward adaptive pitch extraction residual coder. A comparison of this figure with Fig. 2.1 reveals the obvious difference that β and T are now computed from $\hat{s}(k)$'s rather than $s(k)$'s. As a result of this change, it is no longer necessary to transmit β and T since, in case of no transmission errors, the receiver can carry out the same computation. Note that the receiver must now be more complex since it must be capable of computing β and T .

The use of $\hat{s}(k)$ to compute β and T causes another less obvious change in the transmitter. In order to use $\hat{s}(k)$'s, the computation of β and T must be based only on past speech estimates. In the forward adaptive case, the computation of β and T is based on both future and past speech samples. The basic approach for BPARC is to compute β and T on a block of $\hat{s}(k)$, $k = k_0, k_0-1, \dots, k_0 - K_c$ and then to use this value of β and T to form reduced speech $v(k)$ for $k = k_0+1, k_0+2, \dots, k_0+K_u$. Here K_c is called the computation block size and K_u is the use block size; these two values are not necessarily equal.

The basic philosophy of the BPARC approach is that in voiced segments, where pitch redundancy removal is most effective, β and T does not change rapidly. Table 6.1 illustrates this fact with a listing of β and T for a typical segment of voiced speech. The use of a β and T , computed for one block, in the next block will not have much effect on the performance in such a voiced segment.

There will, however, be rapid changes in β and T during transitions from voiced to unvoiced or unvoiced to voiced speech. During these transitions, significant performance degradation can be expected in BPARC

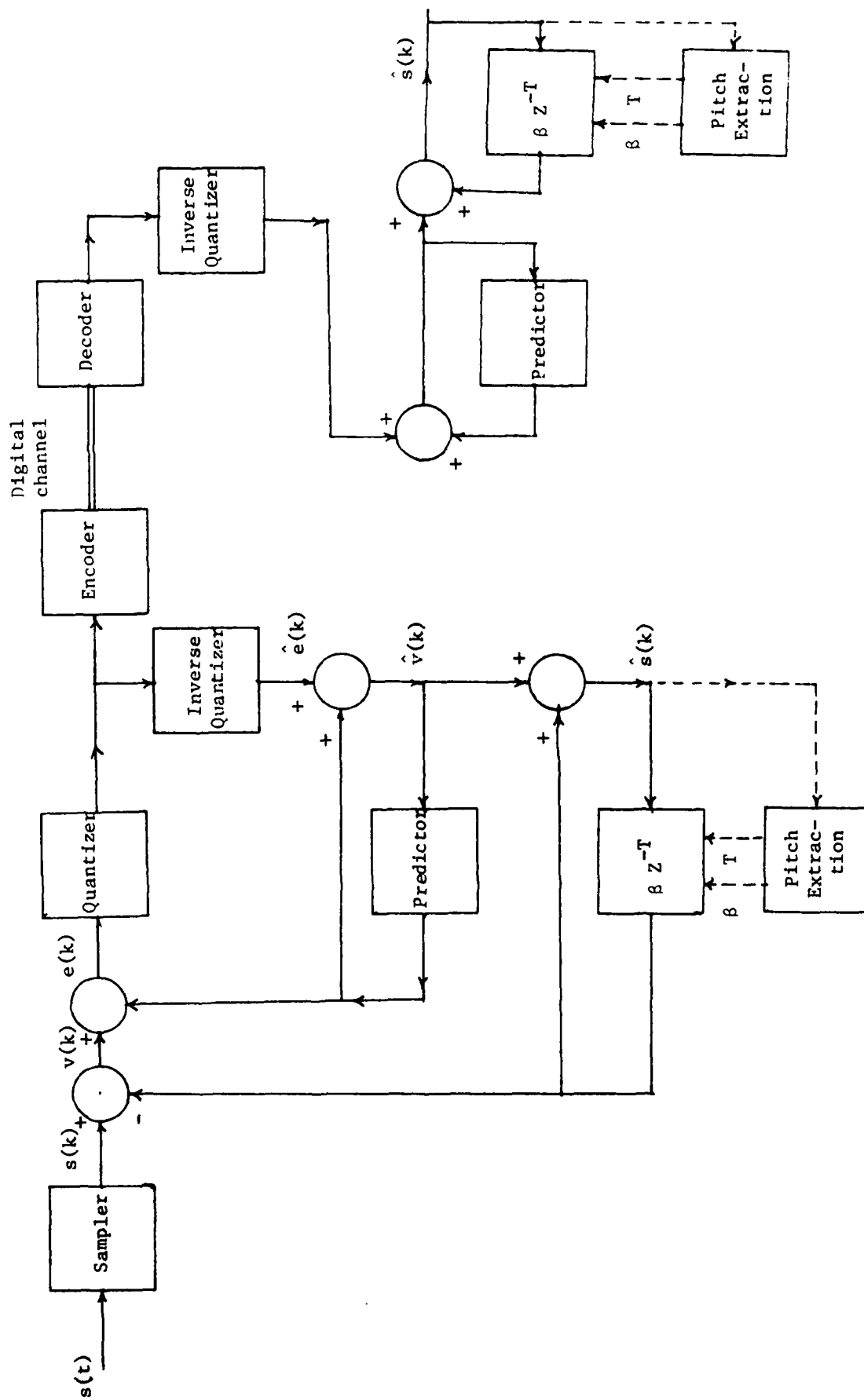


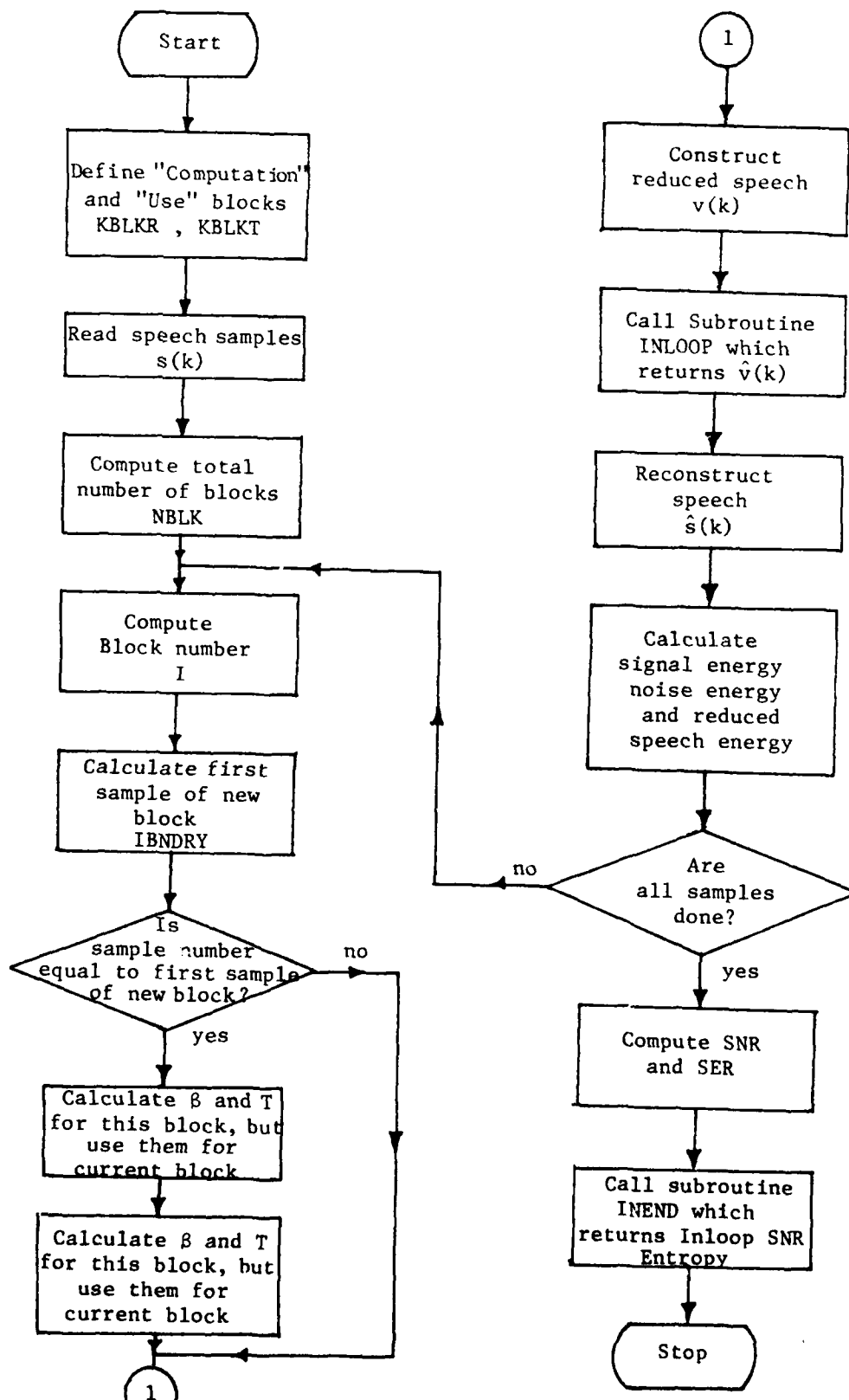
Fig. 6.1 Backward Pitch Extraction Adaptive Residual Coder

Table 6.1

Values of β and T for a Segment of Voiced Speech

| Sample Number | β | T |
|---------------|---------|----|
| 5400 | 0.91 | 30 |
| 5520 | 0.80 | 30 |
| 5550 | 0.86 | 30 |
| 5500 | 0.92 | 30 |
| 5610 | 0.99 | 30 |
| 5640 | 0.95 | 30 |
| 5670 | 1.04 | 30 |
| 5700 | 0.98 | 30 |
| 5730 | 0.98 | 30 |
| 5760 | 1.02 | 30 |
| 5790 | 1.03 | 30 |
| 5020 | 1.00 | 30 |
| 5850 | 1.02 | 30 |
| 5880 | 1.00 | 30 |
| 5910 | 1.06 | 30 |
| 5940 | 1.06 | 30 |
| 5970 | 1.04 | 30 |
| 6000 | 1.09 | 30 |
| 6030 | 1.08 | 30 |
| 6060 | 1.09 | 30 |
| 6090 | 1.05 | 30 |
| 6120 | 1.06 | 30 |
| 6150 | 1.06 | 30 |
| 6180 | 1.10 | 30 |
| 6210 | 1.06 | 30 |
| 6240 | 1.06 | 30 |
| 6270 | 1.03 | 30 |
| 6300 | 1.02 | 30 |
| 6330 | 1.02 | 30 |
| 6360 | 1.00 | 30 |
| 6390 | 1.00 | 30 |
| 6420 | 0.98 | 30 |
| 6450 | 0.99 | 30 |
| 6480 | 0.97 | 30 |
| 6510 | 0.95 | 30 |
| 6540 | 0.93 | 30 |
| 6570 | 0.94 | 30 |
| 6600 | 0.94 | 30 |
| 6630 | 0.87 | 30 |
| 6660 | 0.89 | 30 |
| 6690 | 0.85 | 30 |
| 6720 | 0.04 | 30 |

Fig. 6.2 Backward Pitch extraction Adaptive
Residual Coder (BPARC)



over the PARC algorithm. The hope was that the decrease in quantization noise caused by removing the β and T transmission would offset this degradation. The advantage of not requiring framing for β and T is also obvious.

Exactly the same method, namely, Eqs. (2.6) and (2.7) are used to compute β and T for the BPARC as for the standard PARC except \hat{s} is used in place of s and all \hat{s} 's are past values. Hence T at stage k is the value of τ which minimizes the AMDF function given by

$$A(\tau) = \frac{1}{K_c} \sum_{j=k-K_c}^k |\hat{s}(j) - \hat{s}(j-\tau)| \quad (6.1)$$

$\tau = 20, 21, \dots, \tau_{\max}$

Once T is known, β is determined from

$$\beta = \frac{\sum_{j=k-K_c}^k \hat{s}(j) \hat{s}(j-T)}{\sum_{j=k-K_c}^k \hat{s}(j-T) \hat{s}(j-T)} \quad (6.2)$$

A block diagram for the complete BPARC algorithm is given in Fig. 6.2. This algorithm assumes that a complete sentence is read in and then processed. This algorithm was programmed on the PDP-11/60 in order to determine its performance characteristics.

6.3 Performance Evaluation and Parametric Studies

As the first step in evaluating the performance of the BPARC algorithm, a comparison of β , T and SER using the normal (forward) PARC and the BPARC algorithms was made. Table 6.2 shows a typical portion of the results from this study. Note that SER is significantly degraded in the transition regions at the beginning and end. There is some decrease (~ 1 db) in SER in the voiced segment. A computation block length of 30 samples was used for both cases. This degradation in SER is understandable since we use β and T which were calculated from previously processed speech are used for the present block of samples. In transition regions such as V/UV, UV/S, S/V, the previously processed speech block can be much different than the current block of speech to be processed.

A study of the effect of different lengths of "computation block" and "use block" on SNR was conducted next. In general for very large "computation" or "use" block SNR goes down while for smaller block lengths increase in performance was noticed. However, there is not a monotonic increase or decrease in performance noticed with a decrease or increase in block lengths. The results are tabulated in Table 6.3.

From the very basic idea of BPARC, it is clear that this approach should work better for voiced speech. Exactly the same thing was observed when the range of possible β 's was limited to correspond to voiced speech. Table 6.4 shows the improvement in SNR by restricting β .

Table 6.4
Effect of Limit Range for β
(Sent 11, Female Speaker, $K_u = K_c = 30$)

| | $-2 < \beta < 2$ | $0.74 < \beta < 1.14$ |
|-------------|------------------|-----------------------|
| SNR | 19.96 db | 20.69 db |
| SNR(inloop) | 15.00 db | 14.28 db |
| SER | 4.96 db | 6.41 db |
| H | 1.51 b/sample | 1.46 b/sample |

Table 6.2
Comparison of β , T and SER for PARC and BPARC (Sent 11)

| Sample number | PARC | | | BPARC | | |
|---------------|---------|----|-------------------------|---------|----|-------------------------|
| | β | T | Signal Energy Reduction | β | T | Signal Energy Reduction |
| 5310 | 0.37 | 29 | 1.11 00 | 0.00 | 23 | 0.00 |
| 5340 | 2.00 | 26 | 4.36 08 | 0.00 | 23 | 0.00 |
| 5370 | 1.32 | 26 | 7.71 08 | 0.00 | 26 | 0.00 |
| 5400 | 0.81 | 28 | 8.90 08 | 0.00 | 26 | 0.00 |
| 5430 | 0.93 | 29 | 13.54 08 | 0.89 | 27 | 2.54 |
| 5460 | 0.90 | 29 | 11.90 08 | 0.85 | 28 | 3.67 |
| 5490 | 0.91 | 30 | 16.55 08 | 0.91 | 29 | 5.05 |
| 5520 | 0.80 | 30 | 15.54 08 | 0.88 | 30 | 14.72 |
| 5550 | 0.86 | 30 | 10.35 08 | 0.87 | 30 | 10.94 |
| 5580 | 0.92 | 30 | 14.90 08 | 0.88 | 30 | 14.94 |
| 5610 | 0.99 | 30 | 15.70 08 | 0.90 | 30 | 15.57 |
| 5640 | 0.95 | 30 | 13.46 08 | 0.95 | 30 | 14.82 |
| 5670 | 1.04 | 30 | 18.42 08 | 0.95 | 30 | 16.37 |
| 5700 | 0.98 | 30 | 16.69 08 | 0.99 | 30 | 16.62 |
| 5730 | 0.98 | 30 | 14.80 08 | 1.01 | 30 | 15.16 |
| 5760 | 1.02 | 30 | 17.09 08 | 0.96 | 30 | 19.33 |
| 5790 | 1.03 | 30 | 17.40 08 | 0.99 | 30 | 20.15 |
| 5820 | 1.00 | 30 | 17.30 08 | 1.03 | 30 | 16.39 |
| 5850 | 1.00 | 30 | 19.08 08 | 1.01 | 30 | 18.69 |
| 5880 | 1.00 | 30 | 22.39 08 | 1.02 | 30 | 19.72 |
| 5910 | 1.06 | 30 | 20.45 08 | 0.99 | 30 | 18.55 |
| 5940 | 1.06 | 30 | 15.61 08 | 1.02 | 30 | 16.58 |
| 5970 | 1.04 | 30 | 18.70 08 | 1.07 | 30 | 18.96 |
| 6000 | 1.09 | 30 | 17.62 08 | 1.07 | 30 | 17.03 |
| 6030 | 1.08 | 30 | 16.16 08 | 1.06 | 30 | 16.07 |
| 6060 | 1.09 | 30 | 16.66 08 | 1.07 | 30 | 19.13 |
| 6090 | 1.05 | 30 | 15.84 08 | 1.09 | 30 | 15.74 |
| 6120 | 1.06 | 30 | 13.82 08 | 1.06 | 30 | 14.31 |
| 6150 | 1.06 | 30 | 11.71 08 | 1.08 | 30 | 12.02 |
| 6180 | 1.10 | 30 | 15.62 08 | 1.08 | 30 | 15.72 |
| 6210 | 1.06 | 30 | 16.46 08 | 1.08 | 30 | 16.42 |
| 6240 | 1.06 | 30 | 14.30 08 | 1.08 | 30 | 13.59 |
| 6270 | 1.03 | 30 | 14.53 08 | 1.05 | 30 | 14.03 |
| 6300 | 1.02 | 30 | 16.95 08 | 1.05 | 30 | 17.52 |
| 6330 | 1.02 | 30 | 17.48 08 | 1.03 | 30 | 16.40 |
| 6360 | 1.00 | 30 | 17.05 08 | 1.01 | 30 | 17.34 |
| 6390 | 1.00 | 30 | 21.79 08 | 1.01 | 30 | 21.29 |
| 6420 | 0.90 | 30 | 17.56 08 | 1.00 | 30 | 18.34 |
| 6450 | 0.99 | 30 | 20.76 08 | 0.99 | 30 | 19.43 |
| 6480 | 0.97 | 30 | 18.49 08 | 0.98 | 30 | 18.26 |
| 6510 | 0.95 | 30 | 10.08 08 | 0.97 | 30 | 19.37 |
| 6540 | 0.93 | 30 | 13.91 08 | 0.96 | 30 | 12.93 |
| 6570 | 0.94 | 30 | 12.06 08 | 0.94 | 30 | 13.94 |
| 6600 | 0.94 | 30 | 12.80 08 | 0.92 | 30 | 12.68 |
| 6630 | 0.87 | 30 | 13.60 08 | 0.95 | 30 | 12.51 |
| 6660 | 0.80 | 30 | 12.98 08 | 0.89 | 30 | 14.22 |
| 6690 | 0.85 | 30 | 10.01 08 | 0.88 | 30 | 10.55 |
| 6720 | 0.84 | 30 | 14.74 08 | 0.87 | 30 | 14.47 |
| 6750 | 0.76 | 30 | 9.9 08 | 0.8 | 30 | 9 |
| 6780 | 0.76 | 30 | 0.72 08 | 0.83 | 30 | 8.3 |

Table 6.3
Variation of Use and Computation Block Lengths
USE BLOCK LENGTH K_u (Samples)

| | | 20 | 30 | 40 | 50 |
|--|----|---------------------------------|--------------------------------|--------------------------------|--------------------------------|
| COMPUTATION BLOCK LENGTH K_c (Samples) | 20 | 19.06* 15.05 4.01 1.53 | 20.05 14.10 5.96 1.51 | | |
| | 30 | 19.23 14.62 4.61 1.51 | 19.96 15.00 4.96 1.51 | 18.80 15.59 3.22 1.50 | |
| | 40 | 19.25 15.11 4.15 1.48 | 19.43 14.82 4.61 1.49 | 19.15 15.27 3.88 1.50 | 19.39 14.76 4.62 1.49 |
| | 50 | 19.46 14.91 4.55 1.49 | 20.06 14.12 5.94 1.50 | 19.22 15.02 4.21 1.50 | 18.34 14.97 3.37 1.50 |
| | 60 | 19.19 14.98 4.21 1.49 | 19.64 14.56 5.08 1.49 | 19.17 15.05 4.11 1.51 | 18.89 15.13 3.75 1.50 |
| | 70 | 19.24 14.90 4.35 1.51 | 19.53 14.93 4.60 1.48 | 19.05 15.61 3.44 1.48 | 18.43 15.00 3.43 1.51 |
| | 80 | | 19.80 14.84 4.96 1.49 | 18.61 15.06 3.55 1.52 | 18.12 14.86 3.26 1.51 |
| | 90 | | | 18.90 15.09 3.81 1.49 | 18.57 15.29 3.28 1.51 |

*Each block lists SNR (overall), SNR (inloop), SER and H.

AD-A086 133

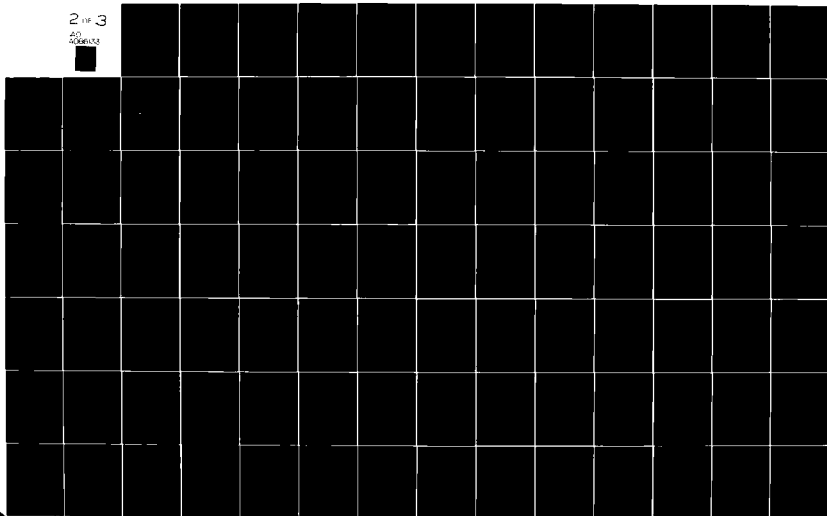
NOTRE DAME UNIV IN DEPT OF ELECTRICAL ENGINEERING F/S 17/2
DESIGN AND IMPLEMENTATION OF A SPEECH CODING ALGORITHM AT 9600 --ETC(U)
APR 80 J L MELSA, D L COHN, A ARORA DCA100-79-C-0005

UNCLASSIFIED

NL

2 OF 3

AD-A086 133



In the above approach, β was set zero if it was not in the allowed range. It appeared that the resulting sharp changes in β would degrade performance. To eliminate this problem a limit was set on the maximum percentage changes in β so that β would vary more smoothly in transition regions. This approach helped to increase SER as shown in Table 6.5 but not the performance as SNR inloop decreased.

Table 6.5
Effect of Limiting $\Delta\beta$
(Sent 1, male speaker, $K_c = 45$, $K_u = 30$), $0.65 \leq \beta \leq 1.25$)

| | $\Delta\beta = 0$ | $ \Delta\beta < 10\%$ | $ \Delta\beta < 20\%$ |
|-------------|-------------------|------------------------|------------------------|
| SNR | 17.78 db | 17.57 db | 17.70 db |
| SNR(inloop) | 14.84 db | 14.45 db | 14.67 db |
| SER | 2.94 db | 3.12 db | 3.02 db |
| H | 1.49 b/sample | 1.50 b/sample | 1.49 b/sample |

To calculate the pitch parameter T the function

$$A(T) = \sum_{j=1}^K |\hat{s}(j) - \hat{s}(j-T)| \quad \text{was}$$

minimized with respect to T for $20 \leq T \leq L$ where $L \geq K_c$ computation block length = 30. Different values of L were tried. Some improvement in performance was noticed (see Table 6.6) if search range was decreased up to a certain point. For a long search range, the effect of errors that were made while reconstructing previous samples becomes more severe while a small search range may not be enough to detect correct period T.

Table 6.6
Effect of Search Range for T
(Sent 1 : male speaker, $K_u = K_c = 30$)

| | $20 \leq T \leq 100$ | $20 \leq T \leq 80$ | $20 \leq T \leq 70$ |
|-------------|----------------------|---------------------|---------------------|
| SNR | 17.67 db | 17.78 db | 17.78 db |
| SNR(inloop) | 14.87 db | 14.84 db | 14.75 db |
| SER | 2.80 db | 2.94 db | 3.03 db |
| H | 1.49 b/sample | 1.49 b/sample | 1.49 b/sample |

As discussed earlier for the normal PARC algorithm it is necessary to transmit β 's and T's along with quantized residual reduced speech. This, of course, requires a few bits per sample. If a block size of 100 is used and β and T require 13 bits, this becomes 0.13 bits/sample. That leaves 1.37 bits/sample for transmission of other information. Hence, to make a fair comparison between the two algorithms, H was limited to 1.37 for PARC and 1.5 bits/sample for BPARC. Even though BPARC is not a clear winner (see Table 6.7), it is a very attractive solution to problems of transmitting β and T.

Table 6.7
Comparison of PARC and BPARC Algorithms

| | | SNR | SNR inloop | SER | H |
|-------------|-------|----------|---------------|---------|---------------|
| Sentence 1 | PARC | 19.11 db | 13.63 db | 5.48 db | 1.37 b/sample |
| | BPARC | 17.78 db | 14.84 db | 2.94 db | 1.49 b/sample |
| Sentence 11 | PARC | 21.40 db | 12.95 db | 8.45 db | 1.36 b/sample |
| | BPARC | 20.53 db | 14.21 db | 6.31 db | 1.47 b/sample |

6.4 Transmission Error Studies

No speech encoding algorithm is good unless it can tolerate with channel errors. To study the effects of channel errors, first step is to determine the effect of errors in quantizer levels. The errors that are introduced in quantizer levels do not exactly correspond to errors caused due to bit reversal but nevertheless they are a good measure of the algorithm's susceptibility to channel errors.

The following procedure was adopted to study effect of channel errors in the BPARC algorithm:

- (i) The receiver program was extracted from transmitter program.
- (ii) A quantizer output file was created with desired transmission errors.
- (iii) This file was read by receiver and a reconstructed speech $\hat{s}(k)$ file was created.
- (iv) SNR was calculated between original speech and received speech.

The algorithm tolerated one transmission error (for male speaker SNR goes down from 19.52 to 19.48) but becomes unstable with additional errors. It was suspected that wrong \hat{s} 's lead to wrong β 's and possibly wrong T's which in turn make \hat{s} 's wrong. To investigate the cause, the BPARC program was run with no pitch removal. The algorithm works very well (see Table 6.8) even with 1% transmission error rate thus confirming above doubt.

Table 6.8
Effect of Transmission Errors on BPARC with no Pitch Extractor
(Sentence 1: Male speaker,
SNR with no transmission error = 18.83 db, (H=2.18))

| Transmission error rate | SNR |
|----------------------------|----------|
| 0.01% | 18.82 db |
| 0.1% | 18.37 db |
| 1% | 4.25 db |

A fixed predictor instead of adaptive predictor was also tried in the system. It was found that fixed predictor minimizes the effect of transmission errors to some extent.

Table 6.9 lists β 's and T's with and without transmission error. Note that β changes immediately after error has been introduced. The plots in Figs. 6.3 and 6.4 compare the effect of transmission error on local SNR.

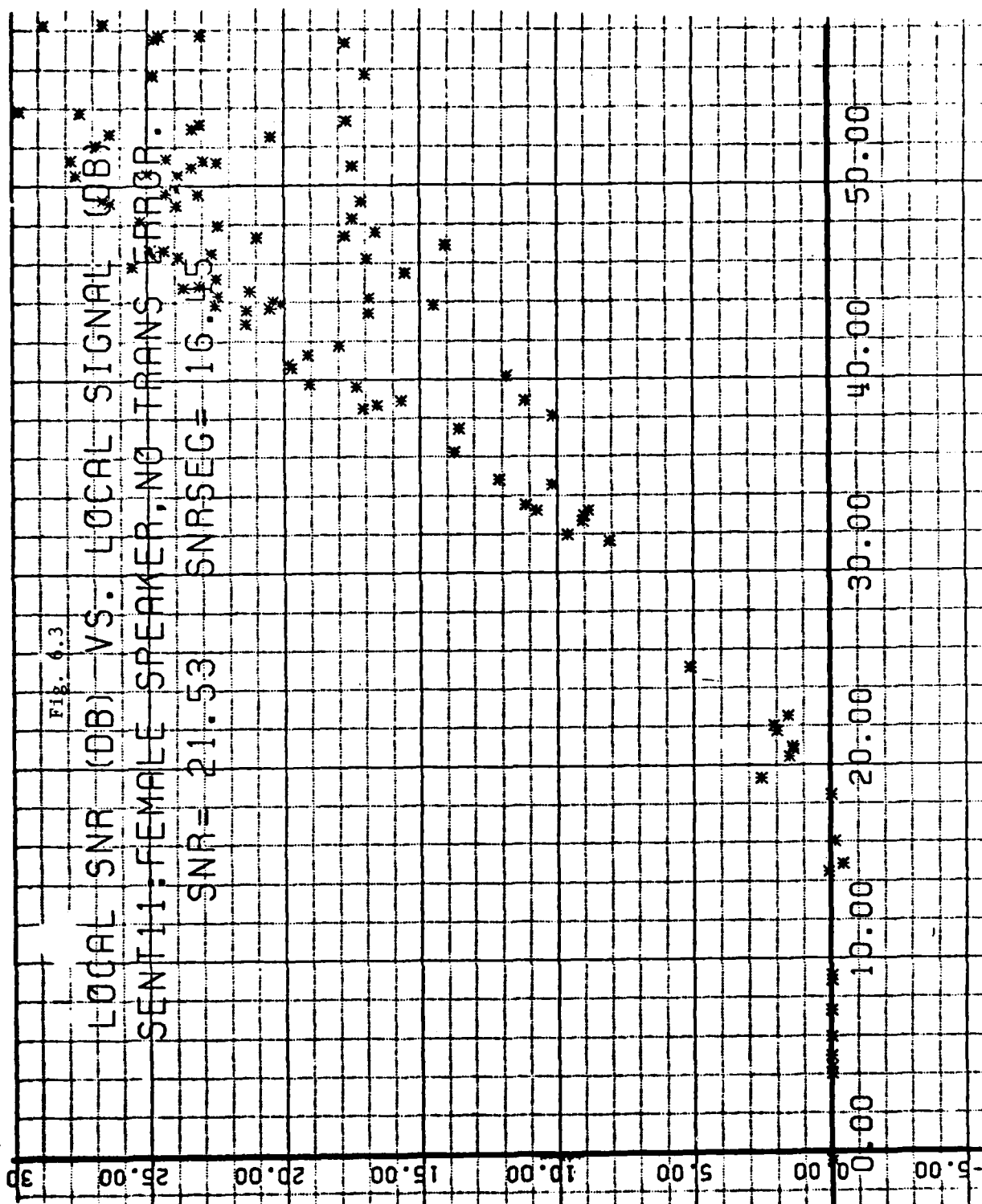
Table 6.9

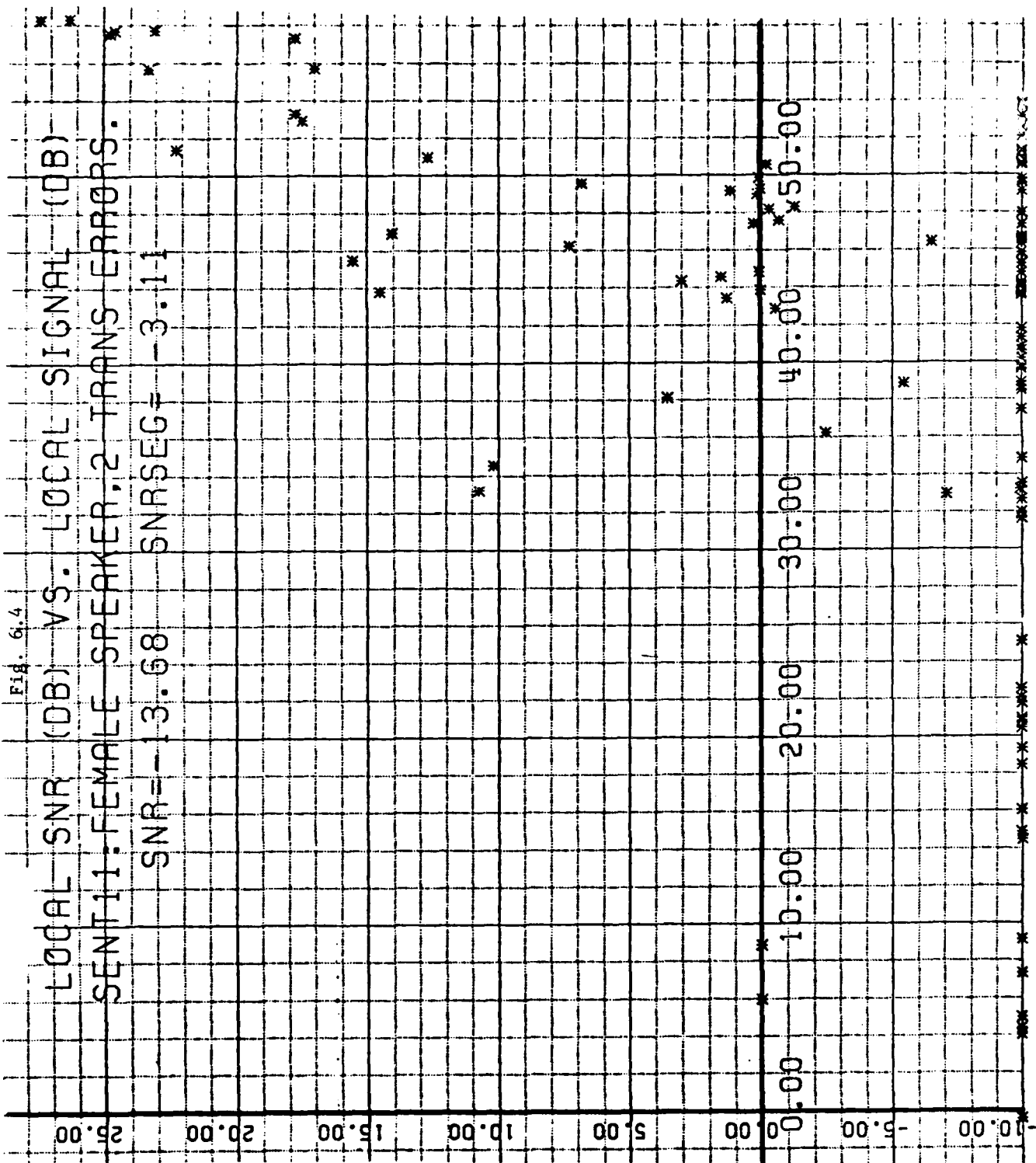
Effect of Transmission Error (Sentence 11: Female speaker)

Without transmission error

With transmission error at
sample number 1674, 7075

| Sample Number | β | T | β | T |
|---------------|---------|----|---------|----|
| 1290 | 0.88 | 27 | 0.88 | 27 |
| 1320 | 0.94 | 27 | 0.94 | 27 |
| 1350 | 0.97 | 55 | 0.97 | 55 |
| 1380 | 0.98 | 27 | 0.98 | 27 |
| 1410 | 0.98 | 27 | 0.98 | 27 |
| 1440 | 0.95 | 27 | 0.95 | 27 |
| 1470 | 0.99 | 27 | 0.99 | 27 |
| 1500 | 0.99 | 27 | 0.99 | 27 |
| 1530 | 0.92 | 27 | 0.92 | 27 |
| 1560 | 0.94 | 27 | 0.94 | 27 |
| 1590 | 0.94 | 27 | 0.94 | 27 |
| 1620 | 0.97 | 55 | 0.97 | 55 |
| 1650 | 0.98 | 55 | 0.98 | 55 |
| 1680 | 1.01 | 55 | 1.01 | 55 |
| 1710 | 1.00 | 55 | 0.99 ← | 55 |
| 1740 | 1.02 | 55 | 1.02 | 55 |
| 1770 | 1.02 | 55 | 1.02 | 55 |
| 1800 | 1.00 | 55 | 0.99 | 55 |
| 1830 | 0.93 | 55 | 0.93 | 55 |
| 1860 | 0.89 | 55 | 0.88 | 55 |
| 1890 | 0.81 | 27 | 0.80 | 27 |
| 1920 | 0.83 | 27 | 0.82 | 27 |
| 1950 | 0.87 | 27 | 0.86 | 27 |
| 1980 | 0.91 | 27 | 0.91 | 27 |
| 2010 | 0.91 | 27 | 0.91 | 27 |
| 2040 | 0.90 | 27 | 0.91 | 27 |
| 2070 | 0.90 | 28 | 0.91 | 28 |
| 2100 | 0.87 | 28 | 0.89 | 28 |
| 2130 | 0.88 | 28 | 0.65 | 28 |
| 2160 | 0.70 | 57 | 1.02 | 57 |
| 2190 | 0.70 | 28 | 1.01 | 57 |
| 2220 | 0.80 | 27 | 0.85 | 57 |
| 2250 | 0.80 | 24 | 0.84 | 57 |
| 2280 | 0.80 | 25 | 0.84 | 57 |
| 2310 | 0.80 | 20 | 0.84 | 57 |
| 2340 | 0.80 | 20 | 0.84 | 57 |
| 2370 | 0.80 | 20 | 0.84 | 57 |
| 2400 | 0.80 | 20 | 0.84 | 57 |
| 2430 | 0.80 | 20 | 0.84 | 57 |
| 2460 | 0.80 | 20 | 0.84 | 57 |
| 2490 | 0.80 | 20 | 0.84 | 57 |
| 2520 | 0.80 | 20 | 0.84 | 57 |
| 2550 | 0.80 | 20 | 0.84 | 57 |
| 2580 | 0.80 | 20 | 0.84 | 57 |
| 2610 | 0.80 | 20 | 0.84 | 57 |
| 2640 | 0.80 | 20 | 0.84 | 57 |
| 2670 | 0.80 | 20 | 0.84 | 57 |
| 2700 | 0.80 | 20 | 0.84 | 57 |
| 2730 | 0.80 | 20 | 0.84 | 57 |





6.5 Conclusions

The study presented in this chapter of the BPARC algorithm shows that it is an attractive approach for a fairly error-free channel but not suitable for channel with at least 0.1% error rate. The following changes in algorithm might help to make it more robust.

1. Currents β and T are calculated for computation block (using received speech) and then used for the current block to construct speech. It is now possible to use the new received speech samples for current block and re-calculate β and T using those samples. These values of β and T should be closer to true values for that block.
2. Maximum changes in β could be limited thus reducing effect of channel error.
3. Errors in pitch periods (in voiced region) due to channel errors could be known by looking at T 's of previous blocks and then could be corrected.

Unfortunately time did not permit an examination of these ideas.

CHAPTER 7

TANDEM OPERATION

7.1 Introduction

One of the requirements of the speech coding algorithm is that it should perform satisfactorily in tandem with a CVSD speech coder operating at a data rate of 16 Kb/s and this tandem configuration should provide speech intelligibility with minimal degradation compared with a single link of CVSD operating at 16 Kb/s. The simulation of this tandem operation was done and results are discussed in Sec. 7.2.

Another requirement of speech coding algorithm is that it should produce intelligible speech under acoustic background noise. The simulation of background noise and the performance of PARC are discussed in Sec. 7.3.

7.2 PARC in Tandem with CVSD

The performance of the PARC system in tandem with CVSD system was studied. This study included passing raw speech through CVSD algorithm to create a file of CVSD output speech and using this speech file as input to the PARC algorithm. The output of the PARC algorithm becomes the output of the tandem system CVSD-PARC as shown in Fig. 7.1a. Similarly the PARC-CVSD tandem connection shown in Fig. 7.1b was also studied. The CVSD algorithm used for the study of tandem operation is described in Appendix D.

The CVSD algorithm used in this study operates on input speech sampled at the rate of 16K samples per second. This means the sampling rates at the input and the output of CVSD must be modified in order to make the tandem connections with PARC which operates on speech sampled at 6.4 KHz. The resampling can be done by using resampling programs listed in Appendix D. The CVSD program has incorporated this resampling program which makes simulation less time consuming.

The performance of these tandem connections are judged by Signal to quantization Noise Ratio and the subjective criterion. Sentence 1, "Cats and Dogs each hate the other", spoken by a male speaker, was used for simulation. Results are reported in Table 7.1 and 7.2.

TABLE 7.1 SNR for PARC,CVSD and their interconnections.

| PARC | CVSD | PARC-CVSD tandem | CVSD-PARC tandem |
|----------|----------|---------------------|---------------------|
| 18.31 db | 11.78 db | 10.80 db | 10.68 db |

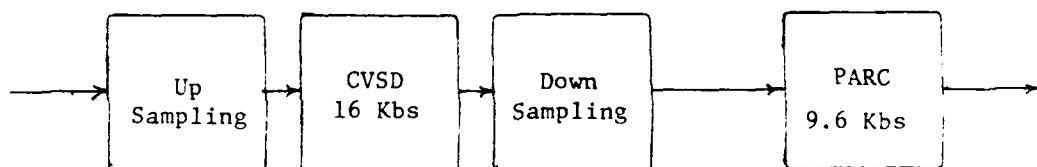


Fig. 7.1a CVSD in Tandem with PARC

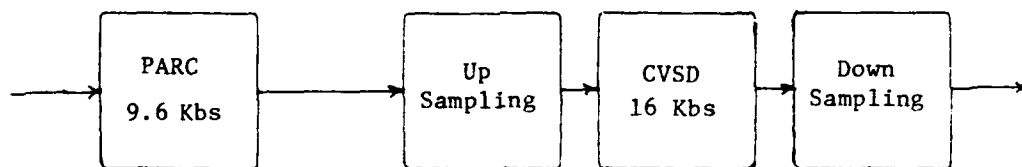


Fig. 7.1b PARC in Tandem with CVSD

TABLE 7.2 Effect of CVSD Speech input
on PARC performance

| | SNR | Inloop SNR | SER | Entropy H |
|-----------------------------------|----------|---------------|---------|------------------|
| PARC with raw speech as input | 18.31 db | 13.69 db | 4.62 db | 1.41 bits/sample |
| PARC with CVSD speech as input | 16.84 db | 12.90 db | 3.94 db | 1.74 bits/sample |

The results in Table 7.1 show that there is little degradation in SNR due to tandem operations. In fact, the speech quality of CVSD-PARC tandem seems to be better than CVSD alone in terms of perception. This study indicated that PARC acts as a filter for the granular noise in the CVSD output. The performance of these tandem operations could be improved by redesigning various parameters used in PARC. However, at this point the purpose of the study was to make sure that PARC algorithm performs reasonably well in tandem with CVSD.

It can be seen from Table 7.2 that the SNR decreased by less than 2 db by inputting the CVSD speech instead of raw speech. This decrease in SNR could be attributed to the decrease in the predictor performance as a result of high frequency contents of CVSD speech and poor signal energy reduction due to decrease in correlation between CVSD speech samples. However, this decrease in SNR is surprisingly low. This shows that the PARC algorithm works very well for noisy input speech except for some increase in entropy. The entropy increase is a rather serious problem but solvable by using buffer control techniques which are discussed in details in Chapter 10.

7.3 Effect of Background Speakers on PARC Performance

It has been observed that PARC algorithm performs satisfactorily well in tandem with CVSD. However, the noise in CVSD speech is white. It is also important to study the effects of correlated background noise such as office noise.

One of the speech coding algorithm requirements is that the speech coder shall produce intelligible speech under conditions of acoustic background noise, 60 db referenced to 20μ Newtons/meter². This statement, though technically precise, gives little feeling about the loudness of noise. Figure 7.2 [1] gives comparative intensities of variety of common sounds. The noise level described above is similar to quiet office noise. Regarding sound energy, Alex ander Woods'[2] quotation gives the whole picture. He, in his book, "Physics of Music", points out that sound energy generated by shouting of the crowd throughout an exciting game (say, 50,000 people at a 90 minute football match between Notre Dame and USC is just about enough to warm one cup of coffee.

The effect of background noise, consisting of typewriter noise, conversation, music and so forth was studied on the real-time system. The algorithm performs with no difficulty.

The study of background noise is very simple after the algorithm has been implemented in real time. It is just a matter of talking into handset with noise in the background. The output could be heard through headphone. However, in the FORTRAN simulation, the task is not so straightforward. There is a need for digital speech file with background noise. It was thought that periodic background noise would be the worst kind of noise for PARC algorithm. Therefore, it was decided to study the performance of the algorithm for multispeaker files. Multispeaker files were created by adding two digital speech files with appropriate weight.

In the simulation, the multispeaker file was generated by adding sentence 11 (female speaker) to male speaker, sentence 7, as shown in Eq. (7.1).

$$s_{\text{composite}} = s_{11} + k s_1 \quad (7.1)$$

where k takes values from 0 to 1 thus having varying degree of background noise. It was noticed that pitch extraction loop picks pitch for both the speakers and algorithm performs very well as can be seen from the results in Table 7.3.

TABLE 7.3 SNR's for Multispeaker Files

| Multispeaker file | SNR | Inloop SNR | SER | Entropy H Bits/sample |
|----------------------|----------|---------------|---------|--------------------------|
| $s_{11} + 0 s_1$ | 21.11 db | 13.01 db | 8.10 db | 1.39 |
| $s_{11} + .25 s_1$ | 20.16 db | 13.03 db | 7.13 db | 1.57 |
| $s_{11} + .5 s_1$ | 19.44 db | 13.39 db | 6.05 db | 1.66 |
| $s_{11} + 1 s_1$ | 18.92 db | 13.95 db | 4.97 db | 1.76 |

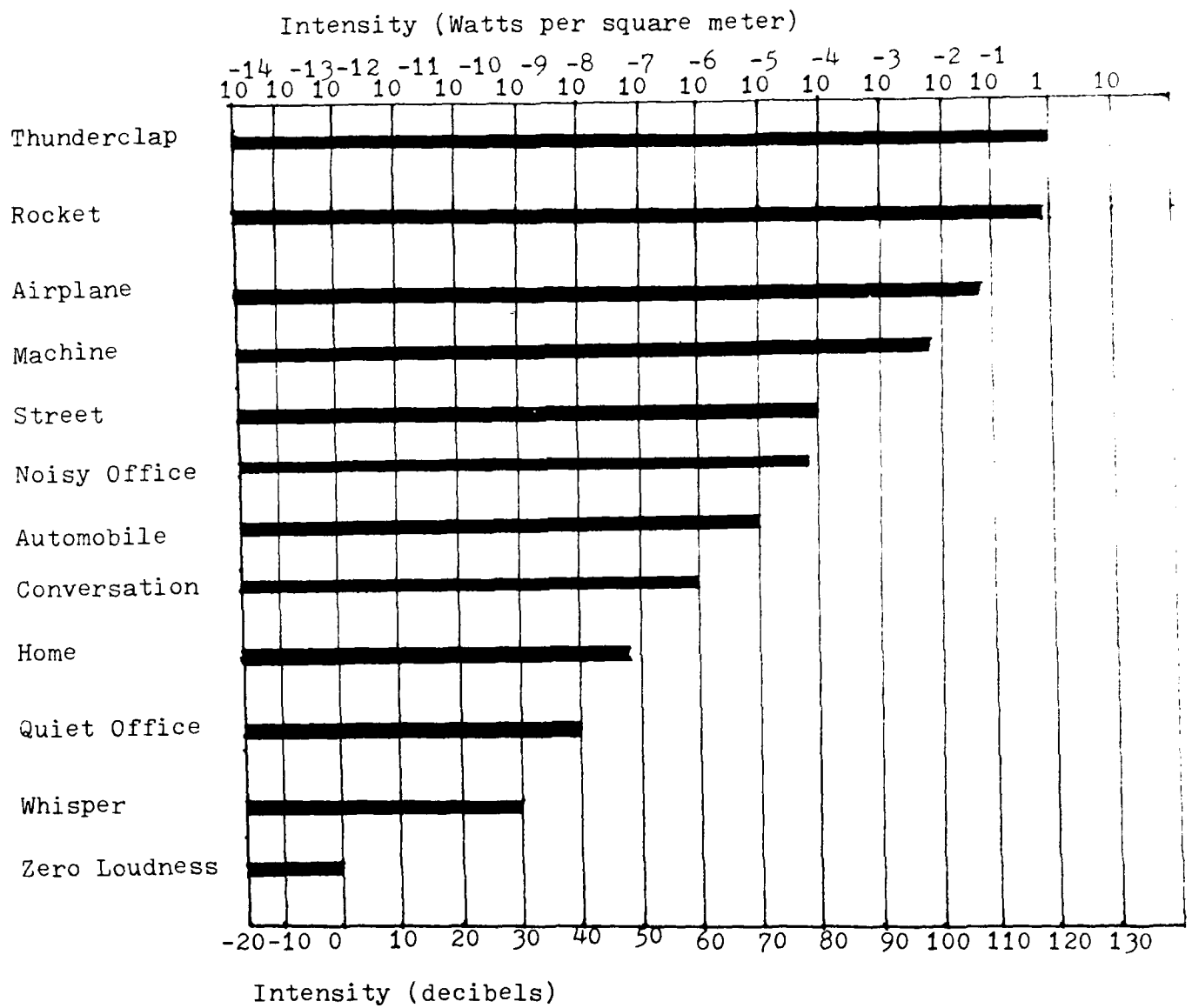


Fig 7.2 Comparative intensities of a variety of common sounds
from bottom to top in order of increasing sound pressure.

7.4 References

- [1] Graham Chedd, " SOUND, from Communication to Noise Pollution",
Doubleday & Company, Inc., New York.
- [2] Wood Alexander, "Physics of Music".

CHAPTER 8

TRANSMISSION ERRORS

8.1 Introduction

Many toll-quality speech links maintain bit-error rates (BER) which are too small (less than 10^{-5}) to affect the quantizer and hence the coder performance. However, a BER of one tenth of a percent is not uncommon and for bad channels this rate could be as high as one percent. In such cases, SNR degradation may be severe unless special precautions are taken. It is important to determine the extent of SNR degradation and if possible how to minimize it.

The study outlined in this chapter is an attempt to answer the question posed above. Section 8.2 describes the method of introducing random transmission errors. This method was simulated on digital computer such that BER could be changed at run time. With the introduction of transmission errors, the effect of various parameters such as predictor order, coarseness of the quantizer and various decay constants on SNR was observed. Simulation results are discussed in Section 8.3.

8.2 Simulation of Transmission Errors:

It is assumed that transmission errors that occur in the digital channel as shown in Fig. 8.1 are random in nature. Hence in the simulation it is important to insure that errors do not occur in bursts. Similarly it is assumed that errors are made by bit reversal and not by bit addition or deletion. From the Fig. 8.1, it may appear that the effect of errors introduced in quantizer levels is similar to errors introduced in bit stream representing quantizer levels provided encoding and decoding operations are carried out correctly. However, it must be kept in mind that samples are not gained or lost if the errors are introduced in quantizer levels while they may be if errors are introduced in bit streams. The advantage of introducing errors in the quantizer levels is that the effect of transmission errors can be measured in terms of degradation of SNR so that the effect of various parameters on transmission errors can be easily evaluated. With errors in the bit stream, SNR loses its meaning since samples may no longer be synchronized. Of course, the transmission errors do affect the bit stream. The following procedure was adopted for simulation of transmission error.

1. Separate programs for PARC transmitter and receiver were written. Program asks for the bit error rate and transmitter produces a file of quantizer levels represented by integer numbers from 1 to 11. Receiver program reads quantizer levels from this file and produces file of reconstructed speech samples.
2. Randomized transmission errors were introduced by using RANDU function available in PDP 11/60 library. Care should be taken to make the seeds large enough for this function so that bursts of errors do not occur in the beginning.

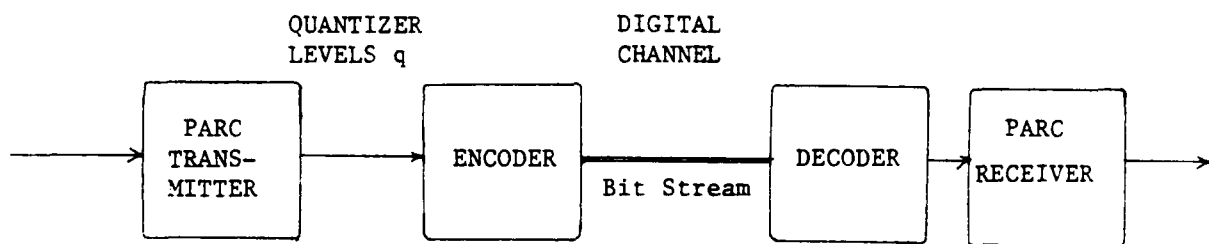


Fig. 8.1 Speech Coder

3. Original speech file and reconstructed speech file is compared and SNR is calculated. This procedure is repeated for different values of parameters.

8.3 Minimizing Transmission Error Effect

Because of error in quantizer level $q(k)$ reconstructed reduced speech $\hat{v}(k)$ and hence reconstructed speech sample $\hat{s}(k)$ is erroneous. This makes $e(k)$ also erroneous which in turn causes a_i 's (predictor parameter) to be incorrect. The effect of $\hat{e}(k)$ on a_i 's can be observed from equation 2. which is repeated here for convenience.

$$a_i(k+1) = a_i(k) + \frac{g \hat{v}(k-1) \hat{e}(k)}{[(1-\alpha) \sum_{j=0}^{\infty} \alpha^j \hat{v}(k-j) + \text{RMSMIN}]}$$

Effect of erroneous $\hat{e}(k)$ on updating a_i 's can be minimized by increasing RMSMIN and decreasing g . This can be seen from Table 8.1 and 8.2.

Table 8.1

Male speaker, sentence 1:

Bit Error Rate 1 in 100

| g | SNR without error | SNR with error |
|-------|----------------------|-------------------|
| 0.015 | 19.25 db | 5.83 db |
| 0.01 | 19.07 db | 6.49 db |

Table 8.2

Male speaker, sentence 1:

Bit Error Rate 1 in 100

| RMSMIN | SNR without error | SNR with error |
|--------|----------------------|-------------------|
| 70 | 18.96 db | 7.18 db |
| 65 | 19.05 db | 6.95 db |
| 55 | 19.30 db | 8.04 db |
| 52 | 19.29 db | 7.73 db |
| 30 | 19.83 db | 3.66 db |

Predictor output is linear combination of past \hat{v} 's and is given by

$$p(k) = \sum_{i=1}^N a_i(k) \hat{v}(k-i)$$

where N is the order of predictor.

For high predictor order effect of transmission error is more since comparatively larger number of incorrect predictor coefficients and larger number of previous incorrect samples contribute to predictor output. Since a_i 's and \hat{v} 's both are used in constructing $p(k)$, effect of transmission errors for increase in predictor order is rather serious. It was observed that by decreasing predictor order from 8 to 4 SNR (for BER of 1 in 100) improved by 3 db.

Various decay constants such as α , exponential decay for RMS value calculation and δ , decay constant for updating predictor parameters do have effect on performance of system with transmission errors. Choice of α controls the effective interval that contributes to the Rms estimate. This interval is larger for syllabic system while it is smaller for instantaneous system. For both extremes, such as large α (syllabic) and small α (instantaneous), SNR decreased. (Ref. Table 8.3).

Decay constant δ is used to update predictor parameters to prevent the transmission errors to propagate. Larger values of δ improve the system performance with error, as can be seen from Table 8.3.

Table 8.3

| Parameters | SNR | |
|------------------|---------------|--------------------|
| | with no error | with 1% error rate |
| $\alpha = 0.97$ | 19.28 db | 6.13 db |
| $= 0.9$ | 18.96 db | 7.18 db |
| $= 0.8$ | 18.89 db | 5.01 db |
| $*\delta = 0.02$ | 19.46 db | overflow |
| $= 0.04$ | 19.21 db | -0.78 db |
| $= 0.06$ | 19.18 db | 1.63 db |

*With all other parameters optimized for good error performance, degradation is not so severe with above values of δ .

Quantizer noise is also an important factor in affecting the performance of the system with transmission errors. Closer the quantizer levels less is the quantization noise and hence less is the effect of transmission errors. Table 8.4 shows that by taking output levels apart SNR has decreased.

Table 8.4

Symmetric quantizer.

Bit error rate 1 in 100.

| Output levels | SNR without error | SNR with error |
|----------------------|----------------------|-------------------|
| 0 1.8 4.25 6.5 8 12 | 20.13 db | 10.18 db |
| 0 1.9 4.5 7.5 10. 12 | 19.42 db | 7.65 db |

To see the effect of various error rates and the effect of errors in different segments of speech, random errors with 1% and 0.1% error rates were added using different random sequences. The study has shown that 0.1% error rate causes little degradation while it is significant for 1% error rate. However, output speech was found to be intelligible in spite of BER of 1%. It was also noticed that if the error occurs in silence segment of speech its effect is negligible. Considering the fact that 40 to 60% of the speech is silence, effect of small BER is not significant as can be seen from Table 8.5

Table 8.5

Case 1: SNR at transmitter = 20.13 db

| | Error Rate 1% | Error Rate 0.1% |
|--------------------|------------------|--------------------|
| Random Sequence #1 | 7.13 db | 15.14 (15 errors) |
| Random Sequence #2 | 8.57 db | 20.00 (7 errors) |
| Random Sequence #3 | 9.78 db | 14.34 (12 errors) |

Case 2: SNR at transmitter = 18.63 db

| | Error Rate 1% | Error Rate 0.1% |
|--------------------|------------------|--------------------|
| Random Sequence #1 | 6.99 db | 16.41 db |
| Random Sequence #2 | 6.00 db | 18.48 db |
| Random Sequence #3 | 8.53 db | 13.72 db |

8.4 Conclusion

PARC algorithm found to perform well in presence of random channel errors as high as one percent error rate. The degree of adaptation in the algorithm and error performance seems to be related. In general, higher adaptation of parameters and more complex system leads to poor performance in presence of channel errors. For example, reducing the predictor order from 8 to 4 improved the error performance significantly.

PARC employs DPCM quantizer, hence it is more tolerant to randomly occurring bit errors from perceptual point of view than systems which employ PCM quantizers (1),(2). This is because error spikes caused in the reconstruction of a PCM waveform (due to wrongly received bit) can have maximum amplitudes which are in the order of peak of input signal while corresponding spike magnitudes in DPCM decoding really related to the peak value of first difference in the input. The consequent greater magnitude of a typical PCM error spike makes it more annoying in spite of the fact that it does not propagate in time. The effect of channel error on the synchronization has not been investigated.

8.5 References

- [1] K. Y. Chang and R. W. Donaldson, "Analysis, Optimization, and Sensitivity Study of PCM Systems Operating on Noisy Communication Channels," IEEE Trans. Commun., vol. COM-20, pp. 330-350, June 1972.
- [2] J. Yan and R. W. Donaldson, "Subjective Effects of Channel Transmission Errors on PCM and DPCM Voice Communications Systems," IEEE Trans. Commun., vol. COM-20, pp. 281-290, June 1972.

CHAPTER 9

FILTERING

9.1 Introduction

Subjective listening tests have indicated that the most objectionable aspect of the speech generated at the PARC receiver is its granular noise due to quantization errors. Since the spectrum of the quantization noise $n_q(k)$ will, in general, be whiter than the speech signal $s(k)$, it appears reasonable to develop a filter for removing part of the quantization noise from $\hat{s}(k)$ and hence improving the speech.

Section 9.3 describes the pre-emphasis of speech to improve the speech quality in terms of perception. Various choices of filters are outlined in Sec. 9.3 and the results are presented in Sec. 9.4.

In the study of pre-emphasis of speech it was observed that low pass filtering does have considerable effect on entropy. This effect is discussed in Sec. 9.5 and simulation results are presented in Sec. 9.6. Design of low-pass Butterworth filter is outlined in the Appendix.

9.2 Evaluation of Pre-emphasis

For auto-correlated signals, such as speech, predictive coding [1,2] is an efficient method of encoding the signal into digital form. In predictive coders, the quantization noise depends on prediction errors; hence, efficient prediction minimizes quantization error. However, in some segments of speech, such as unvoiced speech, the degree of correlation is small and prediction is therefore poor, resulting in more quantization noise. When the amplitude of this noise is comparable to the speech signal it mars the quality of the received speech. To reduce this problem, it appeared that some sort of pre-emphasis of speech would be helpful.

The basic concept of a pre-emphasis filter as shown in Fig. 9.1 is to spread energy in the input signal over the full bandwidth of the processor. Since most of the energy in speech is in the lower end of the spectrum, the filter is a high-pass filter; and hence, the de-emphasis filter is a low pass filter. As pre-emphasis filter is a high frequency filter, the unvoiced segment of speech gets emphasized. However, the filter design is such that overall energy gain for typical phonetically balanced sentence is approximately to unity.

From the figure, it is clear that pitch extraction is to be carried out on high pass filtered speech to get β 's and T's. However, it was observed that it makes little difference if the β 's and T's are obtained by pitch extraction on original speech.

It might be possible to manipulate the block diagram of PARC by moving filters inside to get the equivalent system. This is of no immediate interest and hence not covered here. However, PARC algorithm

could be modified to include adaptive noise spectral shaping as proposed by Makhoul & Berouti [3].

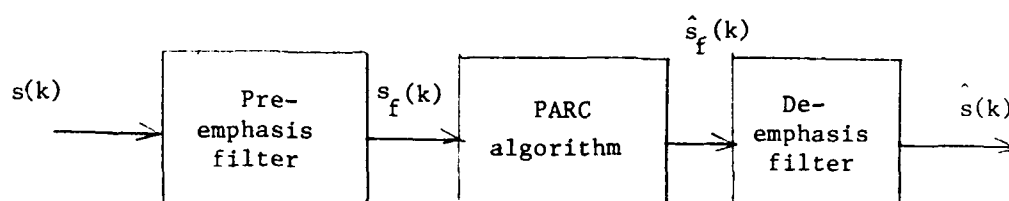


Fig. 9.1 Pre and De-emphasis Filters with PARC Algorithm

9.3 Filter Selection

As mentioned earlier, the pre-emphasis filter is a high-pass filter and de-emphasis is just the inverse of pre-emphasis filter.

In order to minimize complexity, the order of filter is kept small. It was decided to use a 3rd-order filter with general form

$$s_f(k) = \frac{1}{K} s(k) + \frac{a}{K} s(k-1) + \frac{b}{K} s(k-2) + \frac{c}{K} s(k-3) \quad (9.1)$$

The Z transform of this filter is

$$H(z) = \frac{1 + aZ^{-1} + bZ^{-2} + cZ^{-3}}{K} \quad (9.2)$$

Then, using the fact that Z is just $e^{j\omega T}$ where T is the sampling interval the system function becomes

$$H(j\omega) = \frac{1 + ae^{-j\omega T} + be^{-j2\omega T} + ce^{-j3\omega T}}{K} \quad (9.3)$$

and the magnitude of the system function is

$$|H(e^{j\omega T})|^2 = \frac{1 + a^2 + b^2 + c^2 + [2a + 2b(a + c)] \cos \omega T + 2(b + ac) \cos 2\omega T + 2c \cos 3\omega T}{K} \quad (9.4)$$

The four filters whose performance will be described in details in the next section are shown in Table 9.1.

Table 9.1
Filter Parameters

| | K | a | b | c |
|----------|--------|--------|-------|--------|
| filter 1 | 0.5113 | -1.182 | 0.677 | -0.140 |
| filter 2 | 0.5718 | -0.888 | 0.486 | -0.095 |
| filter 3 | 0.6718 | -0.626 | 0.347 | -0.064 |
| filter 4 | 0.7311 | -0.508 | 0.219 | -0.053 |

The frequency response of these filters is shown in Fig. 9.2 for a sampling rate of 6.4 kss.

The de-emphasis filter is just the inverse of the pre-emphasis filter. Therefore, for filter of Eq. (9.3) the inverse is

$$\hat{s}(k) = K s_f(k) - a\hat{s}(k-1) - b\hat{s}(k-2) = c\hat{s}(k-3) \quad (9.5)$$

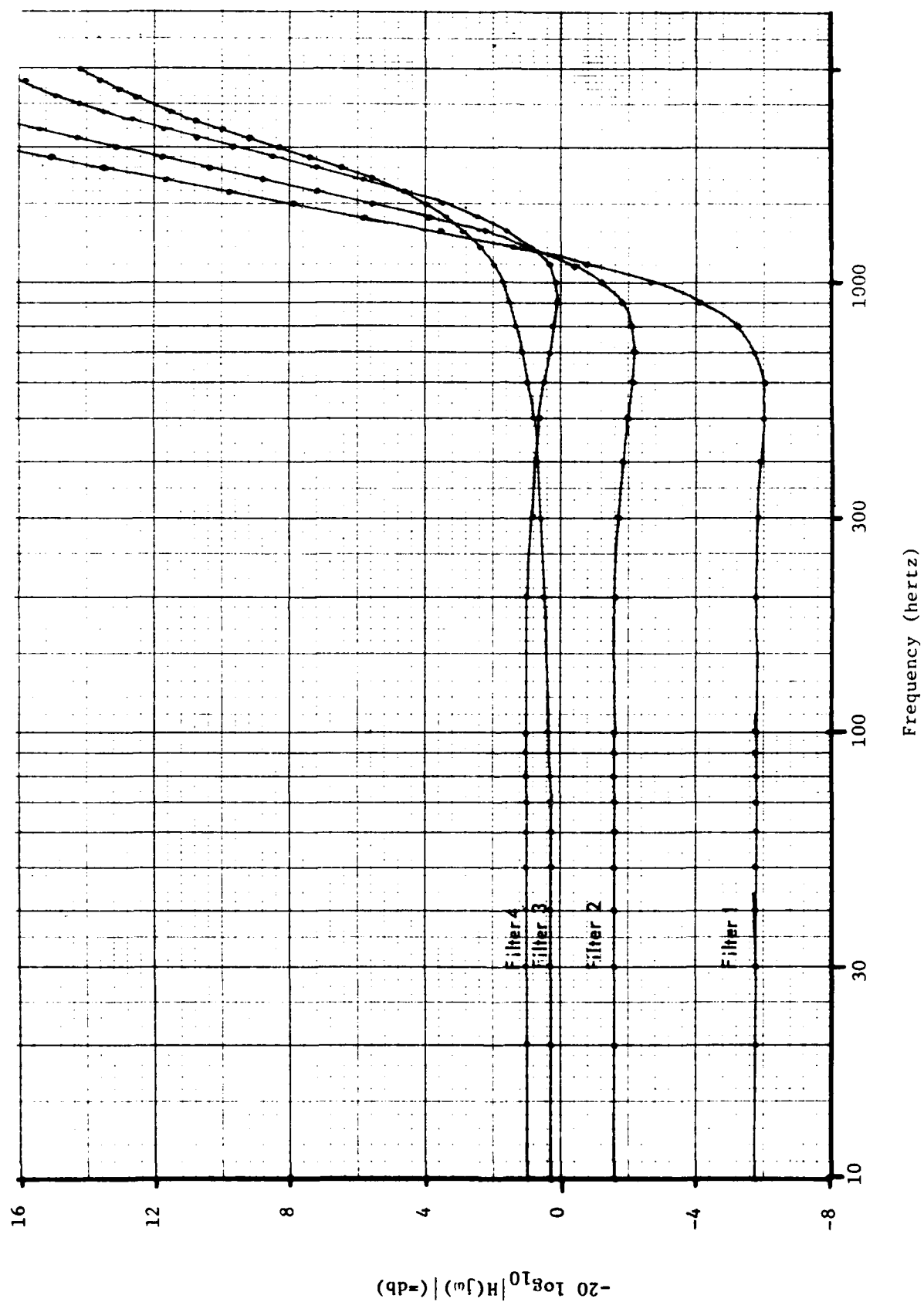


Fig. 9.2 Frequency Response of Pre-emphasis Filters

9.4 Results

The four pre-emphasis filters shown in Fig. 9.2 were evaluated on a PARC operating in the 9.6 kbs mode. Sentence 1, "Cats and Dogs each hate the other" was used for the simulation. Following parameters were computed to evaluate the performance of the filters.

$$\text{SNR(PARC)} = \frac{\sum s_f^2(k)}{\sum (s_f(k) - \hat{s}_f(k))^2} \quad (9.6)$$

$$\text{SNR(overall)} = \frac{\sum s^2(k)}{\sum (s(k) - \hat{s}(k))^2} \quad (9.7)$$

$$\text{SEGSNR} = \frac{\sum_{i=1}^n \text{SNR}_i}{n} \quad (9.8)$$

where n - number of blocks of block length 120 samples in this case.

$$\text{SER (Signal Energy Reduction)} = -10 \log \frac{\sum v_f^2(k)}{\sum s_f^2(k)} \quad (9.9)$$

$$\text{where } v_f(k) = s_f(k) - \beta \hat{s}_f(k-T)$$

It was noticed that by pre-emphasizing speech, the output speech is perceptually better than without pre-emphasis. However, there is an increase in entropy value. This is due to the fact that the increased amplitude of high frequency speech generates more upper levels of quantizer thus generating more bits. All results are reported in Table 9.2.

Table 9.2
Signal to Noise Ratios
with Various Pre-emphasis Filters

| | Product of Pre & De-emphasis Filter Gain | SNR PARC | SER | SNR Whole System | SEGSNR | Entropy H bits/sec |
|------------|--|-------------|---------|------------------------|----------|--------------------------|
| no filters | - | 18.54 db | 5.43 db | 18.54 db | 11.02 db | 1.50 |
| filter 1 | 1.003 | 15.63 db | 2.71 db | 16.73 db | 9.68 db | 1.97 |
| filter 2 | 0.994 | 16.44 db | 3.58 db | 18.64 db | 11.02 db | 1.88 |
| filter 3 | 0.999 | 17.41 db | 4.44 db | 19.36 db | 11.65 db | 1.74 |
| filter 4 | 0.997 | 17.69 db | 4.73 db | 19.20 db | 11.55 db | 1.69 |

9.5 Low-pass Filtering vs. Entropy

For the 9.6 Kbs transmission rate and 6.4 KHz sampling frequency, the number of bits per sample is 1.5. Transmission of parameters such as β and T take a few bits per sample. Therefore, the entropy in the simulation must be maintained at the value less than 1.5. In the previous section, it was observed that pre-emphasis makes the output speech perceptually better; however, it also increases entropy which is unacceptable. To overcome this problem one could use coarser quantization to make the entropy small to begin with and then employ pre- and de-emphasis filters. Unfortunately, the improvement in speech quality due to pre-emphasis operation is not significant enough to consider the above approach.

Another method for achieving good speech quality while controlling the bit rate would be to select parameters such that the speech quality is excellent disregarding the increase in entropy and using the buffer control to check the bit rate. How this buffer control works is discussed in details in the next chapter. The use of a filtering operation to control the bit rate is discussed here.

If a low pass filter is used instead of high pass filter as a pre-emphasis filter, energy reduction is improved and as a result entropy drops. Therefore, the buffer will fill at a slower rate and buffer control would be used infrequently; consequently, there would be less degradation caused by the use of buffer control. However, low pass filtering with bandwidth less than 3200 Hz causes some loss of speech naturalness. It was noticed that during high energy, voiced segment, of speech that the bit rate is higher and hence the buffer fills faster. If the bit rate is brought down by employing low-pass filtering after buffer content is greater than a particular threshold, a double purpose is served. One, the buffer filling

operation is slowed down thus avoiding or delaying the drastic buffer control operation. Second, high frequency componets in low energy unvoiced segments of speech are not filtered out since filter is in operation only after particular threshold thus preserving natural quality. This threshold was found by plotting the buffer content against time for a typical sentence and noting the value of buffer content for voiced speech. For the bit buffer in the FORTRAN simulation of the PARC algorithm, 500 bits appeared to be reasonable a threshold value.

As the cut-off frequency of low-pass filter is decreased, the performance of the predictor improves thus decreasing bit rate further. Thus, the filter cut-off frequency can be decreased depending how full the buffer gets. Hence the pre-emphasis filter is now an adaptive low-pass filter, adaptation of the cut-off frequency depending on the buffer contents.

The low-pass filter used is a simple 3rd order Butterworth filter. Its design and frequency plots are given in an appendix to still chapter.

9.6 Results and Conclusion

The FORTRAN simulation of PARC was modified to include the adaptive low-pass filter concept. To insure that different filters are not employed for every sample of speech when the buffer contents are close to the threshold, a hysteresis structure was utilized. Once a particular filter is selected, it employed in the algorithm for next block of 100 samples. The buffer content is compared with the thresholds only after the block of speech samples is processed. This is shown in the flow chart in Fig. 9.3. The simulation was carried out for Sentence 1: male speaker. The effects of low-pass filtering on entropy and signal energy reduction are tabulated in Table 9.5 while effects on buffer content and bit rates are reported in Table 9.6.

Table 9.5
Effects of Low pass Filtering

| | SNR | SER | Entropy H |
|-----------------------------|----------|--------|------------------|
| no filter | 14.85 db | 4.1 db | 1.37 bits/sample |
| LPF with cut-off 1400 Hz | 16.43 db | 5.1 db | 1.95 bits/sample |

Table 9.6
Effect of LPF on Bit Rate

| | Sample Number | Change in Buffer Content | Increase Bit/ Sample | # Bit Rate |
|------------------------------|------------------|-----------------------------|-------------------------|---------------|
| no filter | 300 - 1000 | 800 | 1.14 | 2.49 |
| LFP with Cut- off 1400 Hz | 300 - 1000 | 680 | 0.97 | 2.32 |

* Bit rate is obtained by adding 1.35 bits/sample to the bit/sample increase in buffer. This is because on an average of 1.35 bits/samples are transmitted on digital channel.

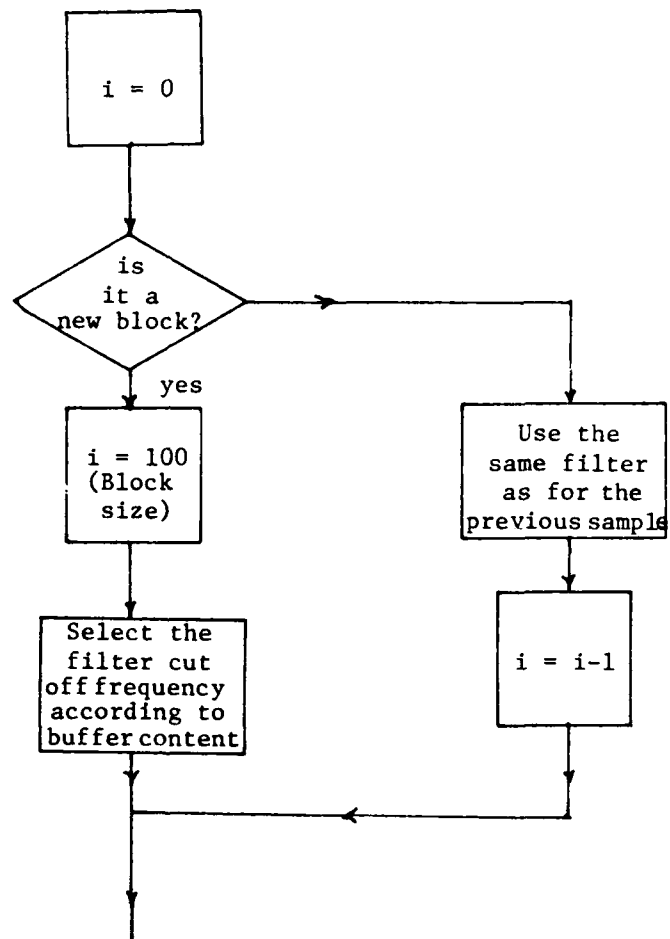


Fig. 9.3 Hysteresis Structure of Adaptive Low Pass Filter

The results in Table 9.5 and 9.6 indicate that Low-pass filtering can be used to control bit rate. In the real-time simulation on the MAP, the structure of programming was such that a varying number of speech samples would be processed each time to get exact predetermined number of bits. This could cause double or triple filtering of the same speech samples. This situation occurs in voiced segments of speech.

The all-pole 3rd order filter could not be used since double and triple filtering causes frequency response to have peaks at the 3 db frequency. This is undesirable and hence there is a need to find new methods of employing adaptive low filtering for buffer control.

It was observed that multiple filtering reduces the 3 db frequency. As mentioned earlier more bits are generated in voiced speech and filter cut-off frequency has to be reduced to cut down the bit rate. Since multiple filter case happens in voiced speech and since it is the region where low-pass cut-off frequency needs to be decreased, the design of single filter would be enough. Thus, there would be no need to change the filter as buffer gets closer to being full. This happens automatically by multiple filtering when buffer gets closer to being full. The filter which gives required change in 3 db frequency upon repetition was designed and design and frequency response is outlined in the appendix.

9.7 References

- [1] P. Elias, "Predictive Coding," IRE Trans. Inform. Theory, vol. IT-1, pp. 16-33, Mar. 1955.
- [2] B. S. Atal and M. R. Schroeder, "Adaptive Predictive Coding of Speech Signals," Bell Syst. Tech. J., vol. 49, pp. 1973-1986, Oct. 1970.
- [3] Makhoul, John and Michael Berouti, "Adaptive Noise Spectral Shaping & Entropy Coding in Predictive Coding of Speech", IEEE Trans. on Acoust. Speech Signal Processing, vol. ASSP-27, No. 1, Feb. 1979, pp. 63-73.
- [4] D. Childers and A. Durling, "Digital Filtering and Signal Processing", West Publishing Company.

9.A Appendix: Derivation of Filters

9.A.1 Butterworth (Maximally flat) Filters

A Butterworth filter is designed to be maximally flat at the origin of the magnitude of the frequency response, i.e., the filter is forced to have as many zero derivatives at the origin of the magnitude response as possible.

The normalized squared magnitude response of the Butterworth filters is

$$|H(\omega)|^2 = \frac{1}{1 + \omega_p^{2n}} \quad (9.8)$$

where $\omega_p = \omega/\omega_c$ is the normalized frequency for an n th order and ω_c is the desired 3 db cut-off frequency of the n th order filter.

In the design of the desired filter frequency response the poles of the transfer function $H(s)$ are needed, then

$$\begin{aligned} H(s)H(-s) &= \frac{1}{1 + (-s^2)^n} \\ &= \begin{cases} 1/(1 + s^{2n}) & n \text{ even} \\ 1/(1 - s^{2n}) & n \text{ odd} \end{cases} \end{aligned} \quad (9.9)$$

Thus, the $2n$ roots of ± 1 are desired depending on the oddness or evenness of the order of the desired filter. Consider the third-order Butterworth filter; for $n=3$, the poles are

$$\begin{aligned} s_0 &= \omega_c \angle 0^\circ & s_{\pm 1} &= \omega_c \angle \pm 60^\circ & s_{\pm 2} &= \omega_c \angle \pm 120^\circ \\ s_3 &= \omega_c \angle 180^\circ \end{aligned}$$

These poles are plotted in Fig. 9.4.

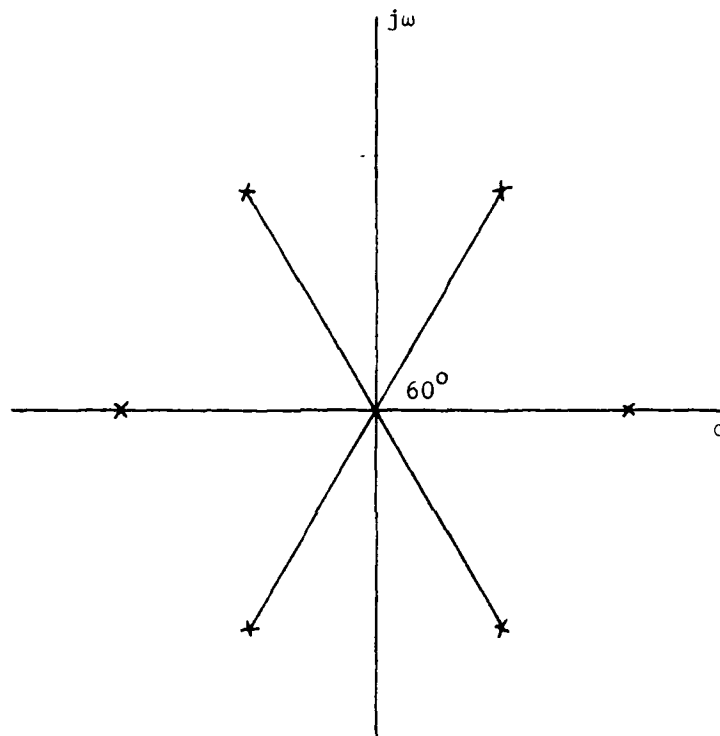


Fig. 9.4 Pole Location of 3rd Order Butterworth Filter

Since the poles of the magnitude squared transfer function are symmetrically placed, the poles which fall in the left half plane are assigned to $H(s)$ for physical realizability.

Thus

$$H(s) = \frac{1}{\prod_{i=1}^3 (s - s_i)} \quad (9.10)$$

where s_i are the left half plane poles of the magnitude squared transfer function. In the third order case

$$s_1 = \left(-\frac{1}{2} - j\frac{\sqrt{3}}{2}\right)\omega_c$$

$$s_2 = \left(-\frac{1}{2} + j\frac{\sqrt{3}}{2}\right)\omega_c$$

$$s_3 = -\omega_c$$

The equivalent transfer function for the digital filter becomes

$$H(z) = \frac{Kz^3}{(z - p_1)(z - p_2)(z - p_3)} \quad (9.11)$$

where

$$p_1 = e^{s_1 T} = \exp \left[\left(-\frac{1}{2} - j\frac{\sqrt{3}}{2}\right)\omega_c T \right]$$

$$p_2 = e^{s_2 T} = \exp \left[\left(-\frac{1}{2} + j\frac{\sqrt{3}}{2}\right)\omega_c T \right]$$

$$p_3 = e^{s_3 T} = \exp [-\omega_c T]$$

$H(z)$ can also be written as

$$H(z) = \frac{K}{1 + az^{-1} + bz^{-2} + cz^{-3}} \quad (9.12)$$

Comparing (9.11) and (9.12)

$$a = -e^{-\frac{\pi f_c}{f_s}} \left[e^{-\frac{\pi f_c}{f_s}} + 2 \cos\left(\frac{\sqrt{3}\pi f_c}{f_s}\right) \right]$$

$$b = -e^{-\frac{2\pi f_c}{f_s}} \left[1 + 2 e^{-\frac{\pi f_c}{f_s}} \cos\left(\frac{\sqrt{3}\pi f_c}{f_s}\right) \right]$$

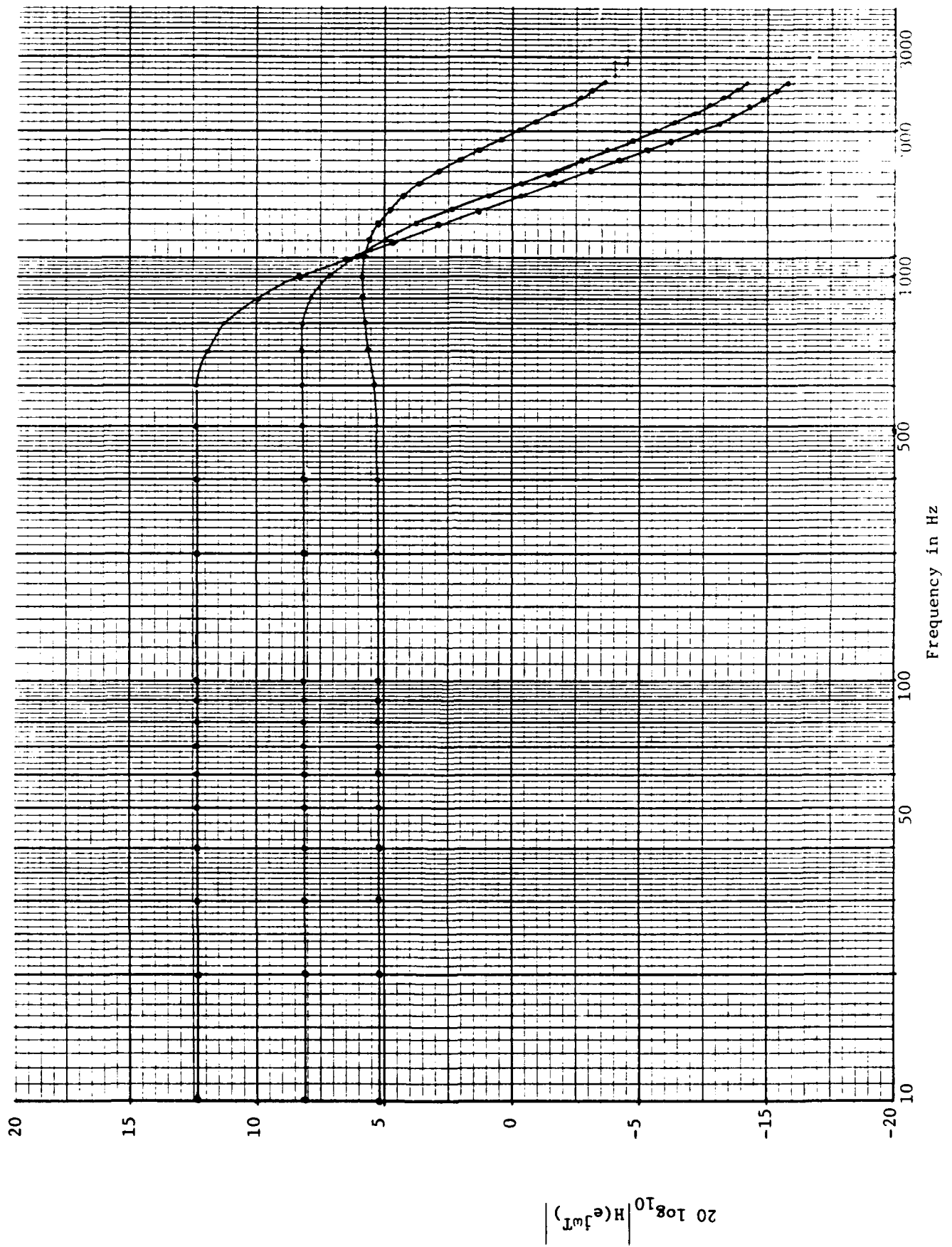
$$c = -e^{-\frac{4\pi f_c}{f_s}}$$

Where f_c - Cut off frequency in Hz.

f_s - Sampling frequency in Hz.

Frequency response of various Butterworth filters is plotted in Fig. 9.5.

Fig. 9. Frequency response of 3rd order Butterworth filter



9.A.2 Design of the low-pass filter used in the algorithm

As mentioned in the earlier sections of this chapter, low pass filtering was seen as a method of softly degrading the speech to avoid overflowing the speech buffer. This low pass filter had to be designed, however, to operate in the MAP.

The first design of the necessary digital filter was done using the impulse invariant transformation on a third order Butterworth filter. Unfortunately, the resulting digital filter had a relatively large amount of ripple which was not desirable, especially because multiple filtering was desired. To assure monotonicity, then, the digital filter was re-designed, using the conformal bilinear transform. Also, it appeared that for ease of implementation in the MAP that the digital filter has only one zero in the z-plane. Thus, the general form of the transfer function of the digital filter was

$$H(z) = \frac{\omega_{CA} + \omega_{CA} z^{-1}}{(\omega_{CA} + 1) + (\omega_{CA} - 1) z^{-1}} \quad (9.13)$$

where

$$\omega_{CA} = \tan \left(\frac{\pi}{6400} f_c \right),$$

$$f_c = \text{filter cutoff frequency (Hz)}.$$

The frequency response of such a filter with an 1800 Hz cutoff is shown in Fig. 9.6. The frequency response for double filtering is also shown. and it can be seen that that cutoff frequency is about 1350 Hz.

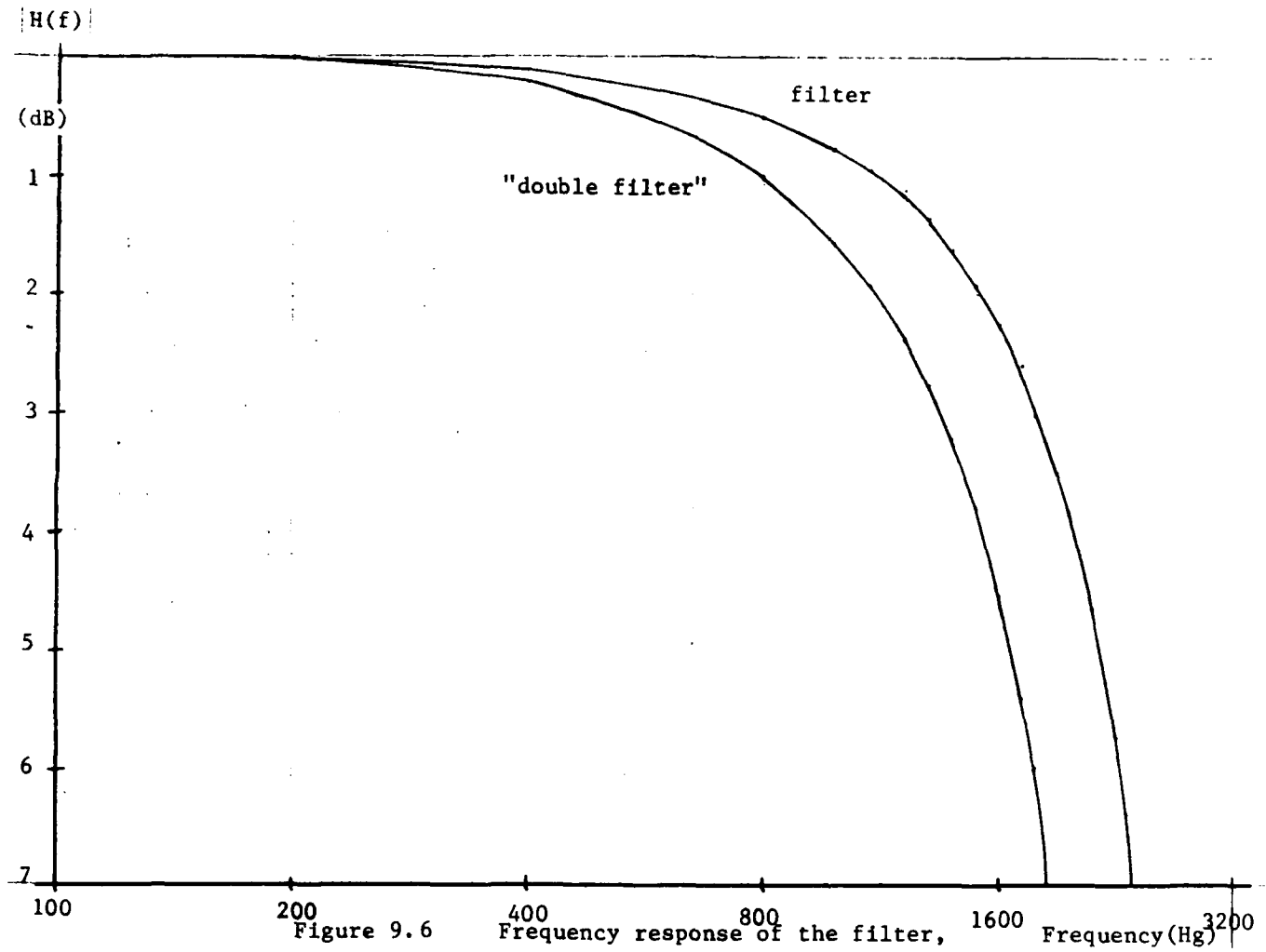


Figure 9.6

Frequency response of the filter,
and the "double filter".

CHAPTER 10

BUFFER CONTROL

10.1 Introduction

Even with the use of adaptive low pass filtering, as described in the previous chapter, to decrease bit-rate generation when the speech buffer approaches overflow, there still remains the ultimate problem of deciding what to do to prevent the buffer from overflowing. Similarly, there is also the problem of deciding what to do to prevent the buffer from underflowing. These two problems are considered in this chapter on buffer control.

10.2 Overflow Control

To prevent buffer overflow, a method had to be found to sharply limit the bit generation rate occasionally which did not cause an unreasonable amount of distortion. Because the quantizer employs feedback and is backward adaptive, it is possible to obtain buffer control by denying the use of certain quantizer levels (or by selectively permitting the use of additional quantizer levels), or by varying the decision thresholds for the quantizer levels. A number of simulation runs were made of both of these techniques without a great deal of success. It appeared that dropping or adding levels was too crude to be an effective control. It was found that either produced a very small change in bit rate generation, or a very pronounced change in bit rate generation. Varying the decision threshold did not appear to be very useful, either, because it typically caused too much distortion.

As a result of these investigations, the problem was approached again from a different, angle. Analysis of simulation results showed that the speech buffer was most prone to overflow during instances of voiced speech. This suggested that pitched repetition might be a useful solution.

Pitched repetition relies on the large amount of correlations between pitch periods of voiced speech. In pitched repetition, samples are generated by duplicating the samples from one pitch period earlier. Due to the large amount of correlation, pitched repetition can typically be carried out for short periods of time during voiced speech, without greatly affecting the subjective quality of speech.

Details of the implementation of pitched repetition are given in Chapter 2.

10.3 Underflow Control

It is just as important to prevent the transmitter sample buffer from underflowing as it is to prevent it from overflowing. This is because when the transmitter sample buffer underflows, the receiver sample buffer overflows. Thus, some method was needed to prevent the transmitter sample buffer from underflowing.

A relatively easy solution to this problem was found by the use of "null" quantizer levels. This technique involves the transmission of a specified bit pattern, just like a normal quantizer level, except that it causes nothing to happen and is discarded at the receiver, with nothing being placed in the receiver sample buffer. Null quantizer levels are used as necessary, then, to prevent the transmitter sample buffer from underflowing.

10.4 Special Considerations at the Receiver

In the error-free condition, it is possible to control both the transmitter and receiver sample buffers by controlling just the transmitter sample buffer. In the presence of errors, however, this is no longer the case. For example, due to an error, it would be possible for the receiver sample buffer to underflow without the transmitter sample buffer overflowing. Some simple rules were developed to handle this situation and to resynchronize the buffers. If the receiver sample buffer overflows, the most recent sample is discarded, since it probably represents silence or near silence. This seems to cause the least distortion, and resynchronizes the buffers. If the receiver sample buffer underflows, a quantizer level "1" is inserted. This again causes a minimum of distortion and resynchronizes the buffers.

10.5 Conclusions and Suggestions for Further Research

It is necessary to provide buffer control to prevent potentially catastrophic conditions due to overflow or underflow. Some relatively simple, but effective, strategies for buffer control have been developed to this end.

There is, of course, room for improvement. For example, the null quantizer levels could also be used to force resynchronization of the transmitter and receiver sample buffers. Even more basic questions exist about the problem of buffer control itself, because it would seem that the quantization system is not as efficient as it could be if it regularly runs into overflow and underflow. A related question is why pitched repetition, which takes few bits to transmit, is so effective at a time when the quantizer is operating at a high bit generation rate.

CHAPTER 11

SOURCE AND ERROR CONTROL CODING

11.1. Introduction

Source and error control coding are the interface between the internal variables of the system and the communications channel. The noiseless source coder performs the first step in generating the bits to be transmitted. Its goal is to try to convey all the necessary information using a minimum of bits. The error control coding is then used to increase the probability that these bits will be received without error. Due to differences in quantity and importance, though, the quantizer levels and the side information are handled in different ways by the source and error control coders.

11.2. Source Coding

11.2.1. Quantizer Levels

In most common quantization systems, a very simple source coding procedure is used: typically there are 2^N quantizer levels used by the system, and each quantizer level is coded as a N-bit binary number. In contrast, PARC uses a more sophisticated source coding procedure in order to increase performance. The source code used to encode the 11 quantizer levels and the "null" level use a variable number of bits to represent the levels, with fewer bits being used for the more common levels. In this way, the quantizer level information can be conveyed very efficiently.

The first attempts at designing a simple variable length source code, however, proved frustrating. Simulations showed that a simple variable length source code tended to cause the sample buffer to fill rapidly during segments of voiced speech. Analysis of the situation showed that the problems appeared to stem from the fact that the quantizer levels were not stationary or independent. Instead, analysis showed that the quantizer levels were better represented by a model where the levels were generated by switching between two sources, one representing the quantizer behavior during voiced segments, and another representing the quantizer behavior during unvoiced and silent segments.

In order to take advantage of this phenomenon, then, a new variable length source code was developed. This new source code was what is known as an overfull source code. Overfull codes are characterized by an ability to encode some sequence in more than one way. For example, in the final source code, a sequence of 14 level-1's could be encoded in either of two ways. This

redundancy would appear, at first glance, to decrease the efficiency of the code, but it, in fact, increases the efficiency of the code. In particular, this redundancy is what allows the source code to perform well with a bimodal source. This was accomplished by designing most of the code using the high-entropy (voiced) statistics, and then adding the long-run codeword based on the low-entropy statistics. In this way, the overfull code performs better than a non-overfull code could.

11.2.2. Side Information

Besides encoding the quantizer levels, the source coder must also encode the side information used in the PARC system. This information consists primarily of the pitch extraction coefficient β and the pitch period T . (Pitched repetition is also signaled by the use of a false β .) This encoding is performed for every frame of samples.

The encoding used is fairly straightforward. The pitch period can only be one of 64 possible integers between 20 and 83, so that it can be represented exactly by 6 bits. The encoding used for the pitch extraction coefficient β , however, is slightly more complicated. The first complication is that the value of β must be quantized because it is a real number. It was determined by simulation, though, that the system was relatively insensitive to the quantization of β , and that using 97 quantization levels, evenly distributed between -2 and 2, appeared to have a negligible effect. The other complication was the signaling of pitched repetition through the use of a false β . The signaling itself could be handled easily by simply assigning it an unused β value. It was felt, however, that it was important that this signal not be mistaken. The encoding used for β , then, used 7 bits, with the all zero pattern reserved for pitched repetition. Error suppression was then provided by not assigning any β quantizer levels to the patterns containing one or two 1's. This left 99 patterns for β quantizer levels, while protecting the pitched repetition signal from single bit errors.

11.3. Error Control Coding

In order to maintain system performance when using a communications channel with a relatively high bit error rate, error control coding is provided for each block of bits. There were several constraints, though, which dictated what kind of error control coding could be used. The blocks were required to be about 200 bits long by the pitch information. The block length also was constrained by synchronization requirements; it was also constrained by the lengths which simple coding schemes require. All of the constraints were satisfied by performing the coding over partial blocks, rather than an entire block at a time. The 189 bit block is divided into three 63 bit frames, so that a single-error-correcting (57, 63) Hamming code may be used. As a result, up to 3 bit errors per block can be corrected, greatly improving the performance of the system in a severe environment.

11.4. Conclusions and Suggestions for Further Research

The source and error control coding described in this chapter allows the system to perform efficiently and cope with severe communications environments. The schemes described were rather simple, as appeared to be required for this implementation. There are, of course, more sophisticated methods which could be studied which might improve performance even further. A major question is how to develop quantization schemes which allocate bits efficiently according to subjective criteria. Another major issue is how to design error protection systems which perform well over a large range of bit error rates.

APPENDIX A

FORTRAN SIMULATION OF ALGORITHM

This appendix presents a listing of the FORTRAN simulation of PARC algorithm. This simulation differs from the real-time algorithm in two ways. First, this algorithm operates on a block containing a fixed number of samples rather than a fixed number of bits. Second, the algorithm described here does not have the adaptive filtering of the input sample sequence.

```
! ! COMMAND FILE TO BUILD THE TASK FOR PARC SIMULATION
! !
! REPEAT/CP/FP, REPEAT/NOSP-REPEAT, PITCH, INSTR5, LPPROC, MOD8, INLOO4, GROW4,
! BUFCON, RANKS4, PRUNES, LEFT, MFILE, SNRVAL, SAVE, OUTBUF, INEND5
! /
! ACTFIL=7
! UNITS=16
! //
```

```

FORTTRAN IV-PLUS V02-51          15:47:32    15-APR-88          PAGE 1
REPEAT.FTN /TR:BLOCKS/VR
0001
PROGRAM REPEAT
PURPOSE:  TO SIMULATE THE REPEAT ALGORITHM.

V.R      DATE      NAME      COMMENTS
0.1      16-JUL-79  J.M. KRESSE  EXISTING SOFTWARE
1.0      17-JUL-79  J.M. KRESSE  BACKED UP
1.1      20-JUL-79  J.M. KRESSE  SPLIT OFF LPPROC, MODB
1.2      22-JUL-79  J.M. KRESSE  OUTPUT BUFFER COUNTER
1.10     31-OCT-79  J.M. KRESSE  REMAINING SAMPLES BUG

THIS IS A SPEECH CODING ALGORITHM FOR DIGITAL TRANSMISSION OF
SPEECH AT A RATE OF 9600 BITS PER SECOND. THE ALGORITHM
COMBINES PITCH EXTRACTION LOOP, PITCH COMPENSATING ADAPTIVE
QUANTIZER, SEQUENTIALLY ADAPTIVE PREDICTOR, MULTIPATH TREE
SEARCHING AND ADAPTIVE SOURCE CODING. ADAPTIVE SOURCE CODING
PART OF THE ALGORITHM WILL BE ADDED TO THIS LATER.

SPEECH SIGNAL IS HIGHLY REDUNDANT. THIS REDUNDANCY IS REMOVED
BY EXTRACTING PITCH BLOCK BY BLOCK AND PITCH REDUCED SIGNAL
IS FORMED. A SEQUENTIALLY ADAPTIVE PREDICTOR USING BACKWARD
ADAPTATION, IS USED TO FORM AN ESTIMATE OF PITCH REDUCED SIGNAL.
THE ERROR IN THIS ESTIMATE IS QUANTIZED BY PITCH COMPENSATING
ADAPTIVE QUANTIZER. THE QUANTIZER HOWEVER USES THE MULTIPATH
TREE SEARCHING ALGORITHM TO DETERMINE ITS OUTPUT. THE RESULTING
QUANTIZED OUTPUT IS CODED USING AN ADAPTIVE SOURCE CODING
PROCEDURE. SOURCE CODE ALSO PERMITS TRANSMISSION OF PITCH
INFORMATION AND SYNCHRONIZING SIGNALS.

FOLLOWING IS THE LIST OF VARIABLES.
S      -ARRAY OF ORIGINAL SPEECH SAMPLES.
V      -ARRAY OF REDUCED SPEECH SAMPLES.
VHAT   -ARRAY OF ESTIMATED PITCH REDUCED SPEECH.
SHAT   -ARRAY OF RECONSTRUCTED SPEECH SAMPLES.
T      -PITCH PERIOD.
BETA   -CORRELATION COEFFICIENTS.
KBLK   -BLOCK SIZE.
NBLK   -NUMBER OF BLOCKS.
L      -DEPTH OF TREE IN TREE SEARCHING ALGORITHM.

```


FORTRAM IV-PLUS V02-51 15:47:32 15-APR-88 PAGE 2
 REPEAT.FTN /TR:BLOCKS/VR

```

C
5502 PARAMETER BUFL=16,IBUF1=256,IBUF2=512,KBLK=150
5503 PARAMETER BUFL=16,IBUF1=256,IBUF2=512,KBLK=150
5504 PARAMETER LUNBUF=7,LUNPRT=6,LUNSH=2,LUNTI=6,MXBLCK=200
5505 PARAMETER NAMLEN=33,NO=.FALSE.,NUL=0,YES=.TRUE.
5506 PARAMETER LUNPRT=6,LUNSH=2,MXBLCK=200
5507 BYTE FNAME(NAMLEN),H(40),ICOMM(40),FNAME2(NAMLEN),FNAME3(NAMLEN)
5508 BYTE FNAME(33),FNAME1(33),FNAME2(33),FNAME3(33),H(40),ICOMM(40)
5509 BYTE IDATE(10),ITIME(9),QPLT
5510 INTEGER NCHAR,PREFIX
5511 INTEGER Q,S,T(200)
5512 LOGICAL BUFL
5513 REAL BUFCNT
5514 DIMENSION BETA(MXBLCK),SHAT(IBUF1),V(IBUF1),VHAT(IBUF1)
5515 DIMENSION BETA(200),SHAT(IBUF1),V(IBUF1),VHAT(IBUF1)
5516 COMMON S(IBUF2)
5517 COMMON /ICNT/ ICNT
5518 COMMON /MBUFF/ MBUFF(BUFL)
5519 COMMON /QPLT/ QPLT
5520 MODB1(INDX1)=IAND(INDX1-1,IBUF1-1)
5521 MODB2(INDX2)=IAND(INDX2-1,IBUF2-1)
5522
C
5523 ASK FOR THE FILENAMES, COMMENTS
5524
5525 WRITE(5,510)
5526 FORMAT(' ENTER THE INPUT FILENAME:')
5527 READ(5,520)FNAME
5528 FORMAT(32A1)
5529 WRITE(5,530)
5530 FORMAT(' ENTER THE PARAMETER POINTER FILENAME:')
5531 READ(5,540)FNAME1
5532 FORMAT(32A1)
5533 WRITE(5,550)
5534 FORMAT(' ENTER THE OUTPUT FILENAME:')
5535 READ(5,560)FNAME2
5536 FORMAT(32A1)
5537 WRITE(5,570)
5538 FORMAT(' ENTER THE OUTPUT HEADER COMMENT:')
5539 READ(5,580)ICOMM
5540 FORMAT(40A1)
5541
C
5542 DO UNTIL Y OR N
5543
C
5544 CONTINUE
5545 WRITE(5,591)
5546 FORMAT(' DO YOU WANT AN EFFECTIVE M PLOT? (Y OR N)')
5547 READ(5,592)QPLT
5548 FORMAT(A1)
5549 IF(.NOT.((QPLT.EQ.'Y').OR.(QPLT.EQ.'N'))).GO TO 590
5550 CONTINUE
5551 IF(QPLT.EQ.'Y').GO TO 594
5552 WRITE(5,593)
5553 FORMAT(' ENTER THE EFFECTIVE M FILENAME:')
5554 READ(5,560)FNAME3
5555

```

```

FORTMAN IV-PLUS V02-51          15:47:32      15-APR-88      PAGE 3
REPEAT.FTM      /TR:BLOCKS/UR

0046      594      CONTINUE
0047      WRITE(LUNTI,595)
0048      595      FORMAT(' IF YOU WANT A FILE OF BUFFER COUNTS, ENTER THE'.
1 . FILENAME.'/
2 . OTHERWISE, HIT RETURN.')
```

```

FORTRAN IV-PLUS V02-51      15:47:32    15-APR-89      PAGE 4
REPEAT.FTN /TR:BLOCKS/VR

C C C C C
OPEN THE INPUT FILE.

C C C C C
OPEN(UNIT=1,TYPE='OLD',READONLY,NAME=FNAME,SHARED)

C C C C C
READ THE HEADER ON SPEECH FILE.

C C C C C
READ(1,10)NSENT,IRATE,NSAMP,IUPPR,ILOWR,NTERMS,H
FORMAT(615,10X,40A1)

C C C C C
PRINT THE HEADER.

C C C C C
CALL DATE(I,DATE)
CALL TIME(I,TIME)
WRITE(6,11)I,DATE,ITIME
FORMAT('DATE: ',10A1,' TIME: ',9A1)
WRITE(6,12)
FORMAT('THE SPEECH FILE HEADER:')
WRITE(6,20)NSENT,IRATE,NSAMP,IUPPR,ILOWR,NTERMS,H
FORMAT(1X,615,10X,40A1)

C C C C C
WRITE(6,70)KBLK,NSAMP
FORMAT('FRAME SIZE=',14,' TOTAL SAMPLES=',16)

C C C C C
WRITE(5,71)
FORMAT('ENTER THE VALUE OF THE MULTIPLYING FACTOR ALPHA:')
READ(5,72)ALPHA
FORMAT(615,7)
WRITE(6,73)ALPHA
FORMAT('ALPHA=',G12.4)

C C C C C

```

```

FORTRAN IV-PLUS V02-51      15:47:32      15-APR-80      PAGE 5
/REPEAT.FTN                /TR:BLOCKS/VR

C
C
C
0003      INITIALIZE S,BETA(1),T(1),IFIRST,IBLK1,NREM,AND NREAD
0004      DO 200 I=1,IBUF2
0005          S(I)=0
0006      CONTINUE
0007      BETA(1)=0.
0008      T(1)=1
0009      IFIRST=1
0010      IBLK1 IS THE BLOCK NUMBER.
0011      IBLK1=1
0012      NREM IS THE NUMBER OF SAMPLES REMAINING IN THE BUFFER.
0013      NREM=0
0014      NREAD IS THE NUMBER OF "COMPLETE BUFFER" READS.
0015      NREAD=NSAMP/IBUF1
0016      WRITE(6,74)
0017      FORMAT('BLOCK',5X,'BETA',6X,'T')
0018
0019      74

```

```

FORTRAN IV-PLUS V02-01      15:47:32   15-APR-88      PAGE 6
REPEAT.FTN      /TR:BLOCKS/WR

0094      IF(INREAD.LE.8)GO TO 218
0095      DO 228 IREAD=1,NREAD
C
C
0096      IF(IAND(IREAD,1).EQ.8)GO TO 221
0097      READ(1,238)(S(ITME2),ITME2=1,IBUF1)
0098      FORMAT(1615)
238      C
C
0099      DO WHILE(MOD82(IFRST+KBLK-1).LE.IBUF1)
C
C
0100      IF(MOD82(IFRST+KBLK-1).GT.IBUF1)
0101      GO TO 248
1      CALL PITCH(BETA(IBLK1),T(IBLK1),
1      MOD82(IFRST))
1      WRITE(6,688)IBLK1,BETA(IBLK1),
1      T(IBLK1)
1      FORMAT(2X,I3,2X,G11.4,I4)
688      IBLK1=IBLK1+1
1      IFRST=MOD82(IFRST+KBLK)
C
C
0102      GO TO 258
0103      CONTINUE
0104      GO TO 268
0105
0106
0107

```

15:47:32 15-APR-88

PAGE 7

```

221      1      READ(1,23B)(S(ITME3),ITME3=IBUF1+1,
C          IBUF2)
C
C      DO WHILE(IFRST+KBLK-1.LE.IBUF2)
20B      IF(IFRST+KBLK-1.GT.IBUF2)GO TO 27B
C          CALL PITCH(BETA(IBLK1),T(IBLK1),
C              IFRST)
C          WRITE(6,6B$)IBLK1,BETA(IBLK1),
C              T(IBLK1)
C          IBLK1=IBLK1+1
C          IFRST=IFRST+KBLK
C          GO TO 28B
C          CONTINUE
C          GO TO 26B
C          CONTINUE
C          CONTINUE
C          NREM=IBUF1-MODB1(IFRST)+1
C          NREM=IAND(IBUF1-MODB1(IFRST)+1,IBUF1-1)
C          GO TO 21B
C          CONTINUE
21B

```



```

FORTRAN IV-PLUS V02-S1          15:47:32    15-APR-80    PAGE 10
REPEAT.FTN /TR:BLOCKS/VR

      C
      C      CLOSE THE SPEECH FILE
      C
      C      CLOSE(UNIT=1)
      C
      C      OPEN THE POINTER FILE FOR PARAMETERS.
      C
      C      OPEN(UNIT=3,NAME=FNAME1,TYPE='OLD',READONLY,SHARED)
      C      READ(3,37#)NRUNS
      C      FORMAT(12)
      C      37#
      C      TYPE ' ' IF YOU WISH TO RUN AGAIN,TYPE 1'
      C      ACCEPT *,IRUN
      C      IF(IRUN.NE.1)GOTO 110
      C      DO 76 IRUNS=1,NRUNS
      C      CALL SUBROUTINE INSTRT WHICH RETURNS VALUE OF L.
      C      CALL INSTRT(L)
      C
      C      REOPEN THE SPEECH FILE AND SKIP OVER THE HEADER.
      C
      C      OPEN(UNIT=1,NAME=FNAME,TYPE='OLD',READONLY,SHARED)
      C      READ(1,40#)
      C      FORMAT(80X)
      C      40#
      C      OPEN THE OUTPUT (SHAT) FILE AND WRITE THE HEADER
      C
      C      OPEN(UNIT=2,NAME=FNAME2,TYPE='NEW',CARRIAGECONTROL='LIST')
      C      WRITE(2,401)INSENT,IRATE,NSAMP,IUPPR,ILOWR,INTERMS,ICOMM
      C      FORMAT(615,10X,40A1)
      C      IF(OPLOT.NE.'Y')GO TO 407
      C      1      OPEN(UNIT=4,NAME=FNAME3,TYPE='NEW',
      C      CARRIAGECONTROL='LIST')
      C      WRITE(4,401)INSENT,IRATE,NSAMP,IUPPR,ILOWR,INTERMS,ICOMM
      C      ICNT=0
      C      407      CONTINUE

```

FORTRAN IV-PLUS V02-51 15:47:32 15-APR-80 PAGE 11
 REPEAT.FTN /TR:BLOCKS/WR

```

C
C      INITIALIZE ES,EN,SHAT,VHAT,S
C
0169      ES=B.
0170      EN=B.
0171      DO 41B ITH1=1,IBUF1
0172          SHAT(ITH1)=B.
0173          VHAT(ITH1)=B.
0174          S(1BUF1+ITH1)=B
0175      CONTINUE
0176      WRITE(6,62B)
0177      FORMAT('SAMPLE',5X,'ES',10X,'EN')
0178      BUFCNT=B.
0179      PREFIX=B

```

11.2
 11.2

```

FORTRAN IV-PLUS V22-51      15:47:32      15-APR-88      PAGE 12
REPEAT.FTN      /TR:BLOCKS/VR

      NREAD2=NSAMP/IBUF1
      IF(NREAD2.LE.8)GO TO 428
      DO 438 IREAD2=1,NREAD2

C
C
C      IF(IREAD2 IS EVEN)GO TO 431
      IF(1AND(IREAD2,1).EQ.8)GO TO 431
      READ(1,448)(S(ITM2),ITM2=1,IBUF1)
      FORMAT(1615)
      DO 458 ITM3=1,IBUF1
        ITM4=(IREAD2-1)*IBUF1+ITM3
        CALL LPPROC(ALPHA,BETA,BUFCNT,
1          BUFFIL,EN,ES,IBUF1,ITM3,ITM4,
2          ITM5,ITM7,KBLK,L,LUNBUF,LUNPRT,
3          LUNSH,MXBLCK,PREFIX,SHAT,T,V,
4          VHAT)
        CALL LPPROC(ALPHA,BETA,EN,ES,
1          IBUF1,ITM3,ITM4,ITM5,KBLK,L,
2          LUNPRT,LUNSH,MXBLCK,SHAT,T,V,
3          VHAT)
        ITM5=ITM4-L+1
        ITM6=MODBI(ITM4)
        ITM7=MODBI(ITM5)
        IBLK2=(ITM4-1)/KBLK+1
        IBLK3=(ITM5-1)/KBLK+1
        CONSTRUCT THE PITCH-REDUCED
        SPEECH.
        V(ITM6)=FLOAT(S(ITM3))-ALPHA
        *BETA(IBLK2)=SHAT(MODBI(ITM3
        -T(IBLK2)))
        CALL INLOOP TO GET Q, VHAT
        CALL INLOOP(V(ITM6),Q,VHAT
        (ITM7))
      1
      2
      3
      4
      5
      6
      7
      8
      9
      10
      11
      12
      13
      14
      15
      16
      17
      18
      19
      20
      21
      22
      23
      24
      25
      26
      27
      28
      29
      30
      31
      32
      33
      34
      35
      36
      37
      38
      39
      40
      41
      42
      43
      44
      45
      46
      47
      48
      49
      50
      51
      52
      53
      54
      55
      56
      57
      58
      59
      60
      61
      62
      63
      64
      65
      66
      67
      68
      69
      70
      71
      72
      73
      74
      75
      76
      77
      78
      79
      80
      81
      82
      83
      84
      85
      86
      87
      88
      89
      90
      91
      92
      93
      94
      95
      96
      97
      98
      99
      100
      101
      102
      103
      104
      105
      106
      107
      108
      109
      110
      111
      112
      113
      114
      115
      116
      117
      118
      119
      120
      121
      122
      123
      124
      125
      126
      127
      128
      129
      130
      131
      132
      133
      134
      135
      136
      137
      138
      139
      140
      141
      142
      143
      144
      145
      146
      147
      148
      149
      150
      151
      152
      153
      154
      155
      156
      157
      158
      159
      160
      161
      162
      163
      164
      165
      166
      167
      168
      169
      170
      171
      172
      173
      174
      175
      176
      177
      178
      179
      180
      181
      182
      183
      184
      185
      186
      187
      188
      189
      190
      191
      192
      193
      194
      195
      196
      197
      198
      199
      200
      201
      202
      203
      204
      205
      206
      207
      208
      209
      210
      211
      212
      213
      214
      215
      216
      217
      218
      219
      220
      221
      222
      223
      224
      225
      226
      227
      228
      229
      230
      231
      232
      233
      234
      235
      236
      237
      238
      239
      240
      241
      242
      243
      244
      245
      246
      247
      248
      249
      250
      251
      252
      253
      254
      255
      256
      257
      258
      259
      260
      261
      262
      263
      264
      265
      266
      267
      268
      269
      270
      271
      272
      273
      274
      275
      276
      277
      278
      279
      280
      281
      282
      283
      284
      285
      286
      287
      288
      289
      290
      291
      292
      293
      294
      295
      296
      297
      298
      299
      300
      301
      302
      303
      304
      305
      306
      307
      308
      309
      310
      311
      312
      313
      314
      315
      316
      317
      318
      319
      320
      321
      322
      323
      324
      325
      326
      327
      328
      329
      330
      331
      332
      333
      334
      335
      336
      337
      338
      339
      340
      341
      342
      343
      344
      345
      346
      347
      348
      349
      350
      351
      352
      353
      354
      355
      356
      357
      358
      359
      360
      361
      362
      363
      364
      365
      366
      367
      368
      369
      370
      371
      372
      373
      374
      375
      376
      377
      378
      379
      380
      381
      382
      383
      384
      385
      386
      387
      388
      389
      390
      391
      392
      393
      394
      395
      396
      397
      398
      399
      400
      401
      402
      403
      404
      405
      406
      407
      408
      409
      410
      411
      412
      413
      414
      415
      416
      417
      418
      419
      420
      421
      422
      423
      424
      425
      426
      427
      428
      429
      430
      431
      432
      433
      434
      435
      436
      437
      438
      439
      440
      441
      442
      443
      444
      445
      446
      447
      448
      449
      450
      451
      452
      453
      454
      455
      456
      457
      458
      459
      460
      461
      462
      463
      464
      465
      466
      467
      468
      469
      470
      471
      472
      473
      474
      475
      476
      477
      478
      479
      480
      481
      482
      483
      484
      485
      486
      487
      488
      489
      490
      491
      492
      493
      494
      495
      496
      497
      498
      499
      500
      501
      502
      503
      504
      505
      506
      507
      508
      509
      510
      511
      512
      513
      514
      515
      516
      517
      518
      519
      520
      521
      522
      523
      524
      525
      526
      527
      528
      529
      530
      531
      532
      533
      534
      535
      536
      537
      538
      539
      540
      541
      542
      543
      544
      545
      546
      547
      548
      549
      550
      551
      552
      553
      554
      555
      556
      557
      558
      559
      560
      561
      562
      563
      564
      565
      566
      567
      568
      569
      570
      571
      572
      573
      574
      575
      576
      577
      578
      579
      580
      581
      582
      583
      584
      585
      586
      587
      588
      589
      590
      591
      592
      593
      594
      595
      596
      597
      598
      599
      600
      601
      602
      603
      604
      605
      606
      607
      608
      609
      610
      611
      612
      613
      614
      615
      616
      617
      618
      619
      620
      621
      622
      623
      624
      625
      626
      627
      628
      629
      630
      631
      632
      633
      634
      635
      636
      637
      638
      639
      640
      641
      642
      643
      644
      645
      646
      647
      648
      649
      650
      651
      652
      653
      654
      655
      656
      657
      658
      659
      660
      661
      662
      663
      664
      665
      666
      667
      668
      669
      670
      671
      672
      673
      674
      675
      676
      677
      678
      679
      680
      681
      682
      683
      684
      685
      686
      687
      688
      689
      690
      691
      692
      693
      694
      695
      696
      697
      698
      699
      700
      701
      702
      703
      704
      705
      706
      707
      708
      709
      710
      711
      712
      713
      714
      715
      716
      717
      718
      719
      720
      721
      722
      723
      724
      725
      726
      727
      728
      729
      730
      731
      732
      733
      734
      735
      736
      737
      738
      739
      740
      741
      742
      743
      744
      745
      746
      747
      748
      749
      750
      751
      752
      753
      754
      755
      756
      757
      758
      759
      760
      761
      762
      763
      764
      765
      766
      767
      768
      769
      770
      771
      772
      773
      774
      775
      776
      777
      778
      779
      780
      781
      782
      783
      784
      785
      786
      787
      788
      789
      790
      791
      792
      793
      794
      795
      796
      797
      798
      799
      800
      801
      802
      803
      804
      805
      806
      807
      808
      809
      810
      811
      812
      813
      814
      815
      816
      817
      818
      819
      820
      821
      822
      823
      824
      825
      826
      827
      828
      829
      830
      831
      832
      833
      834
      835
      836
      837
      838
      839
      840
      841
      842
      843
      844
      845
      846
      847
      848
      849
      850
      851
      852
      853
      854
      855
      856
      857
      858
      859
      860
      861
      862
      863
      864
      865
      866
      867
      868
      869
      870
      871
      872
      873
      874
      875
      876
      877
      878
      879
      880
      881
      882
      883
      884
      885
      886
      887
      888
      889
      890
      891
      892
      893
      894
      895
      896
      897
      898
      899
      900
      901
      902
      903
      904
      905
      906
      907
      908
      909
      910
      911
      912
      913
      914
      915
      916
      917
      918
      919
      920
      921
      922
      923
      924
      925
      926
      927
      928
      929
      930
      931
      932
      933
      934
      935
      936
      937
      938
      939
      940
      941
      942
      943
      944
      945
      946
      947
      948
      949
      950
      951
      952
      953
      954
      955
      956
      957
      958
      959
      960
      961
      962
      963
      964
      965
      966
      967
      968
      969
      970
      971
      972
      973
      974
      975
      976
      977
      978
      979
      980
      981
      982
      983
      984
      985
      986
      987
      988
      989
      990
      991
      992
      993
      994
      995
      996
      997
      998
      999
      1000
      1001
      1002
      1003
      1004
      1005
      1006
      1007
      1008
      1009
      1010
      1011
      1012
      1013
      1014
      1015
      1016
      1017
      1018
      1019
      1020
      1021
      1022
      1023
      1024
      1025
      1026
      1027
      1028
      1029
      1030
      1031
      1032
      1033
      1034
      1035
      1036
      1037
      1038
      1039
      1040
      1041
      1042
      1043
      1044
      1045
      1046
      1047
      1048
      1049
      1050
      1051
      1052
      1053
      1054
      1055
      1056
      1057
      1058
      1059
      1060
      1061
      1062
      1063
      1064
      1065
      1066
      1067
      1068
      1069
      1070
      1071
      1072
      1073
      1074
      1075
      1076
      1077
      1078
      1079
      1080
      1081
      1082
      1083
      1084
      1085
      1086
      1087
      1088
      1089
      1090
      1091
      1092
      1093
      1094
      1095
      1096
      1097
      1098
      1099
      1100
      1101
      1102
      1103
      1104
      1105
      1106
      1107
      1108
      1109
      1110
      1111
      1112
      1113
      1114
      1115
      1116
      1117
      1118
      1119
      1120
      1121
      1122
      1123
      1124
      1125
      1126
      1127
      1128
      1129
      1130
      1131
      1132
      1133
      1134
      1135
      1136
      1137
      1138
      1139
      1140
      1141
      1142
      1143
      1144
      1145
      1146
      1147
      1148
      1149
      1150
      1151
      1152
      1153
      1154
      1155
      1156
      1157
      1158
      1159
      1160
      1161
      1162
      1163
      1164
      1165
      1166
      1167
      1168
      1169
      1170
      1171
      1172
      1173
      1174
      1175
      1176
      1177
      1178
      1179
      1180
      1181
      1182
      1183
      1184
      1185
      1186
      1187
      1188
      1189
      1190
      1191
      1192
      1193
      1194
      1195
      1196
      1197
      1198
      1199
      1200
      1201
      1202
      1203
      1204
      1205
      1206
      1207
      1208
      1209
      1210
      1211
      1212
      1213
      1214
      1215
      1216
      1217
      1218
      1219
      1220
      1221
      1222
      1223
      1224
      1225
      1226
      1227
      1228
      1229
      1230
      1231
      1232
      1233
      1234
      1235
      1236
      1237
      1238
      1239
      1240
      1241
      1242
      1243
      1244
      1245
      1246
      1247
      1248
      1249
      1250
      1251
      1252
      1253
      1254
      1255
      1256
      1257
      1258
      1259
      1260
      1261
      1262
      1263
      1264
      1265
      1266
      1267
      1268
      1269
      1270
      1271
      1272
      1273
      1274
      1275
      1276
      1277
      1278
      1279
      1280
      1281
      1282
      1283
      1284
      1285
      1286
      1287
      1288
      1289
      1290
      1291
      1292
      1293
      1294
      1295
      1296
      1297
      1298
      1299
      1300
      1301
      1302
      1303
      1304
      1305
      1306
      1307
      1308
      1309
      1310
      1311
      1312
      1313
      1314
      1315
      1316
      1317
      1318
      1319
      1320
      1321
      1322
      1323
      1324
      1325
      1326
      1327
      1328
      1329
      1330
      1331
      1332
      1333
      1334
      1335
      1336
      1337
      1338
      1339
      1340
      1341
      1342
      1343
      1344
      1345
      1346
      1347
      1348
      1349
      1350
      1351
      1352
      1353
      1354
      1355
      1356
      1357
      1358
      1359
      1360
      1361
      1362
      1363
      1364
      1365
      1366
      1367
      1368
      1369
      1370
      1371
      1372
      1373
      1374
      1375
      1376
      1377
      1378
      1379
      1380
      1381
      1382
      1383
      1384
      1385
      1386
      1387
      1388
      1389
      1390
      1391
      1392
      1393
      1394
      1395
      1396
      1397
      1398
      1399
      1400
      1401
      1402
      1403
      1404
      1405
      1406
      1407
      1408
      1409
      1410
      1411
      1412
      1413
      1414
      1415
      1416
      1417
      1418
      1419
      1420
      1421
      1422
      1423
      1424
      1425
      1426
      1427
      1428
      1429
      1430
      1431
      1432
      1433
      1434
      1435
      1436
      1437
      1438
      1439
      1440
      1441
      1442
      1443
      1444
      1445
      1446
      1447
      1448
      1449
      1450
      1451
      1452
      1453
      1454
      1455
      1456
      1457
      1458
      1459
      1460
      1461
      1462
      1463
      1464
      1465
      1466
      1467
      1468
      1469
      1470
      1471
      1472
      1473
      1474
      1475
      1476
      1477
      1478
      1479
      1480
      1481
      1482
      1483
      1484
      1485
      1486
      1487
      1488
      1489
      1490
      1491
      1492
      1493
      1494
      1495
      1496
      1497
      1498
      1499
      1500
      1501
      1502
      1503
      1504
      1505
      1506
      1507
      1508
      1509
      1510
      1511
      1512
      1513
      1514
      1515
      1516
      1517
      1518
      1519
      1520
      1521
      1522
      1523
      1524
      1525
      1526
      1527
      1528
      1529
      1530
      1531
      1532
      1533
      1534
      1535
      1536
      1537
      1538
      1539
      1540
      1541
      1542
      1543
      1544
      1545
      1546
      1547
      1548
      1549
      1550
      1551
      1552
      1553
      1554
      1555
      1556
      1557
      1558
      1559
      1560
      1561
      1562
      1563
      1564
      1565
      1566
      1567
      1568
      1569
      1570
      1571
      1572
      1573
      1574
      1575
      1576
      1577
      1578
      1579
      1580
      1581
      1582
      1583
      1584
      1585
      1586
      1587
      1588
      1589
      1590
      1591
      1592
      1593
      1594
      1595
      1596
      1597
      1598
      1599
      1600
      1601
      1602
      1603
      1604
      1605
      1606
      1607
      1608
      1609
      1610
      1611
      1612
      1613
      1614
      1615
      1616
      1617
      1618
      1619
      1620
      1621
      1622
      1623
      1624
      1625
      1626
      1627
      1628
      1629
      1630
      1631
      1632
      1633
      1634
      1635
      1636
      1637
      1638
      1639
      1640
      1641
      1642
      1643
      1644
      1645
      1646
      1647
      1648
      1649
      1650
      1651
      1652
      1653
      1654
      1655
      1656
      1657
      1658
      1659
      1660
      1661
      1662
      1663
      1664
      1665
      1666
      1667
      1668
      1669
      1670
      1671
      1672
      1673
      1674
      1675
      1676
      1677
      1678
      1679
      1680
      1681
      1682
      1683
      1684
      1685
      1686
      1687
      1688
      1689
      1690
      1691
      1692
      1693
      1694
      1695
      1696
      1697
      1698
      1699
      1700
      1701
      1702
      1703
      1704
      1705
      1706
      1707
      1708
      1709
      1710
      1711
      1712
      1713
      1714
      1715
      1716
      1717
      1718
      1719
      1720
      1721
      1722
      1723
      1724
      1725
      1726
      1727
      1728
      1729
      1730
      1731
      1732
      1733
      1734
      1735
      1736
      1737
      1738
      1739
      1740
      1741
      1742
      1743
      1744
      1745
      1746
      1747
      1748
      1749
      1750
      1751
      1752
      1753
      1754
      1755
      1756
      1757
      1758
      1759
      1760
      1761
      1762
      1763
      1764
      1765
      1766
      1767
      1768
      1769
      1770
      1771
      1772
      1773
      1774
      1775
      1776
      1777
      1778
      1779
      1780
      1781
      1782
      1783
      1784
      1785
      1786
      1787
      1788
      1789
      1790
      1791
      1792
      1793
      1794
      1795
      1796
      1797
      1798
      1799
      1800
      1801
      1802
      1803
      1804
      1805
      1806
      1807
      1808
      1809
      1810
      1811
      1812
      1813
      1814
      1815
      1816
      1817
      1818
      1819
      1820
      1821
      1822
      1823
      1824
      1825
      1826
      1827
      1828
      1829
      1830
      1831
      1832
      1833
      1834
      1835
      1836
      1837
      1838
      1839
      1840
      1841
      1842
      1843
      1844
      1845
```


15:47:32 15-APR-88

PAGE 14

[illegible]

FORTAN IV-PLUS V02-51
REPEAT.FTN /TR:BLOCKS/VR

15:47:32 15-APR-88

PAGE 15

```

191 READ(1,440)(S(ITM2),ITM2=IBUF1+1,IBUF2)
192 DO 471 ITM3=IBUF1+1,IBUF2
193   ITM4=(I*PRECAD2-2)*IBUF1+ITM3
194   CALL LPPROC(ALPHA,BETA,BUFCNT,
1   BUFFIL,EN,ES,IBUF1,ITM3,ITM4,
2   ITM5,ITM7,KBLK,L,LUNBUF,LUNPRT,
3   LUNSH,MXBLOCK,PREFIX,SHAT,T,V,
4   VHAT)
5   CALL LPPROC(ALPHA,BETA,EN,ES,
6   IBUF1,ITM3,ITM4,ITM5,KBLK,L,
7   LUNPRT,LUNSH,MXBLOCK,SHAT,T,V,
8   VHAT)
9   ITM5=ITM4-L+1
10  ITM6=MODBI(ITM4)
11  ITM7=MODBI(ITM5)
12  IBLK2=(ITM4-1)/KBLK+1
13  IBLK3=(ITM5-1)/KBLK+1
14  V(ITM6)=FLOAT(S(ITM3))-ALPHA
15  *BETA(IBLK2)*SHAT(MODBI(ITM3)
16  -T(IBLK2)))
17  CALL INLOOP(V(ITM6),Q
18  ,VHAT(ITM7))
19  SHAT(ITM7)=VHAT(ITM7)+ALPHA
20  *BETA(IBLK3)*SHAT(MODBI(ITM7)
21  -T(IBLK3)))
22  SHAT(ITM7)=AMAX1(SHAT(ITM7),
23  -2048.)
24  SHAT(ITM7)=AMIN1(SHAT(ITM7),
25  2047.)
26  IF(ITM7.EQ.IBUF1)WRITE(2,474)
27  (NINT(SHAT(IX2)),IX2=1,IBUF1)
28  FSI=FLOAT(S(MODBI(ITM5)))
29  ES=ES+FSI**2
30  EN=EN+(FSI-SHAT(ITM7))**2
31  IF(MOD(ITM5,1000).EQ.0)WRITE(6,
32  510)ITM5,ES,EN
33  CONTINUE
34  GO TO 432
35  CONTINUE
36  GO TO 420
37  CONTINUE
38  471
39  432
40  430
41  420

```

[illegible]

FORTRAM IV-PLUS V82-51
REPEAT.FTN /TR:BLOCKS/WR

15:47:32 15-APR-88

PAGE 17

```

C      1      IF((ITM7.EQ.IBUF1).AND.(ITM5.GT.8))WRITE(2,474) 1-1.1
C      1      (NINT(SHAT(IX3)),IX3=1,IBUF1) 1-1.1
C      1      FSI=FLOAT(S(MODB2(ITM5))) 1-1.1
C      1      ES=ES+FS1**2 1-1.1
C      1      EN=EN+(FS1-SHAT(ITM7))**2 1-1.1
C      1      IF(MOD(ITM5,1888).EQ.8)WRITE(6,618)ITM5,ES,EN 1-1.1
C      1      CONTINUE
C      488      GO TO 481
C      472      READ(1,448)(S(ITM2),ITM2=IBUF1+1,IBUF1+NREAD3)
C      472      DO 482 ITM3=IBUF1+1,IBUF1+NREAD3
C      472      ITM4=(NREAD2-1)=IBUF1+ITM3
C      472      CALL LPPROC(ALPHA,BETA,BUFCNT,BUFFIL,EN,ES, 11.2
C      472      IBUF1,ITM3,ITM4,ITM5,ITM7,KBLK,L,LUNBUF,LUNPRT, 11.2
C      472      LUNSH,MXBLOCK,PREFIX,SHAT,T,V,VHAT)
C      472      CALL LPPROC(ALPHA,BETA,EN,ES,IBUF1,ITM3,ITM4, 1-1.2
C      472      ITM5,KBLK,L,LUNPRT,LUNSH,MXBLOCK,SHAT,T,V,VHAT) 1-1.1
C      472      ITM5=ITM4-L+1 1-1.1
C      472      ITM6=MOD81(ITM4) 1-1.1
C      472      ITM7=MOD81(ITM5) 1-1.1
C      472      IBLK2=(ITM4-1)/KBLK+1 1-1.1
C      472      IBLK3=(ITM5-1)/KBLK+1 1-1.1
C      472      V(ITM6)=FLOAT(S(ITM3))-ALPHA*BETA(IBLK2) 1-1.1
C      472      *SHAT(MODB1(ITM3)-T(IBLK2))) 1-1.1
C      472      CALL INLOOP(V(ITM6),Q,VHAT(ITM7)) 1-1.1
C      472      SHAT(ITM7)=VHAT(ITM7)+ALPHA*BETA(IBLK3) 1-1.1
C      472      *SHAT(MODB1(ITM7)-T(IBLK3))) 1-1.1
C      472      SHAT(ITM7)=AMAX1(SHAT(ITM7),-2848.) 1-1.1
C      472      SHAT(ITM7)=AMIN1(SHAT(ITM7),2847.) 1-1.1
C      472      IF(ITM7.EQ.IBUF1)WRITE(2,474)(NINT(SHAT(IX4)), 1-1.1
C      472      IX4=1,IBUF1) 1-1.1
C      472      FSI=FLOAT(S(MODB2(ITM5))) 1-1.1
C      472      ES=ES+FS1**2 1-1.1
C      472      EN=EN+(FS1-SHAT(ITM7))**2 1-1.1
C      472      IF(MOD(ITM5,1888).EQ.8)WRITE(6,618)ITM5,ES,EN 1-1.1
C      482      CONTINUE
C      481      GO TO 481
C      481      CONTINUE

```

8214
8215
8216


```

FORTRAN IV-PLUS V02-51      15:47:32      15-APR-88      PAGE 18
REPEAT.FTN

      C      CLOSE THE SPEECH FILE.
      C
      C      CLOSE(UNIT=1)
      C
      C      FINISH AND CLOSE THE OUTPUT FILE
      C
      IF(L.EQ.1)GO TO 484
      DO 485 IX5=1,1-1
      ITM8=MODBI(ITMS+IX5)
      SHAT(ITM8)=B.
      IF(ITM8.EQ.IBUF1)WRITE(2,474)(NINT(SHAT(IX6)),
      1      IX6=1,IBUF1)
      FORMAT(16I5)
      CONTINUE
      GO TO 486
      ITM8=ITM7
      GO TO 486
      CONTINUE
      IF(ITM8.NE.IBUF1)WRITE(2,474)(NINT(SHAT(IX7)),IX7=1,ITM8)
      CLOSE(UNIT=2)
      IF(.NOT.((OPLOT.EQ.'Y').AND.(ICNT.NE.B)))GO TO 488
      WRITE(4,487)(MBUFF(IX8),IX8=1,ICNT)
      FORMAT(<ICNT>15)
      CONTINUE
      IF(OPLOT.NE.'Y')GO TO 489
      CLOSE(UNIT=4)
      CONTINUE
      IF(BUFFIL.EQV.NO)GO TO 49B
      CLOSE(UNIT=LUNBUF)
      GO TO 49B
      CONTINUE
      COMPUTE SIGNAL TO NOISE RATIO.
      C
      C      WRITE(6,61B)NSAMP,ES,EN
      C      FORMAT(17,2G12.4)
      C      SNR=1B.*ALOG1B(ES/EN)
      C
      C
      C      WRITE(6,9B) SNR
      C      FORMAT('SNR OVERALL= ',G11.4,' DB')
      C      WRITE(5,9B)SNR
      C      CALL INEND
      C

```

FORTRAN IV-PLUS V02-51 15:47:32 15-APR-80 PAGE 19
 REPEAT.FTN /TR:BLOCKS/VR

```

      C      GOTO 75
      C 118  STOP
      8249  76  CONTINUE
      C
      C      CLOSE THE POINTER FILE FOR PARAMETERS.
      C
      C      CLOSE(UNIT=3)
      8251  CALL DATE(IDATE)
      8252  CALL TIME(ITIME)
      8253  WRITE(6,11)IDATE,ITIME
      8254  STOP
      8255  END
  
```

```

FORTRAN IV-PLUS V82-51          15:32:32    15-APR-88          PAGE 1
PITCH.FTN      /TR:BLOCKS/WR
0001  SUBROUTINE PITCH(BETA,T,IJ)
C
C      PURPOSE:  TO CALCULATE THE PITCH PERIOD T AND THE PREDICTION
C                COEFFICIENT BETA.
C
C      V.R      DATE      NAME      COMMENTS
C      0.1      16-JUL-79    J.M. KRESSE  EXISTING SOFTWARE
C      1.0      17-JUL-79    J.M. KRESSE  BACKED UP
C
C      THIS IS A PITCH EXTRACTION PROGRAM.
C      BASIC PURPOSE OF PITCH EXTRACTION LOOP IN OUR ALGORITHM IS
C      TO REMOVE PITCH REDUNDANCY OF SPEECH SIGNAL. HERE THE
C      BLOCK ADAPTATION IS ASSUMED WITH BLOCK LENGTH K.
C      K VARIES BETWEEN
C      80 AND 200.
C
C      PARAMETER IBUF2=512,ITL=20,ITU=150,KBLK=150
C      PARAMETER IBUF2=512,ITL=20,ITU=150,KBLK=100
C      INTEGER S,T
C      COMMON S(IBUF2)
C      DIMENSION A(ITU),B(ITU)

```

1-1.7
11.7

0002
0003
0004
0005

```

0006      MODB2(INDX1)=IAND(INDX1-1,IBUF2-1)+1
          C
          C
          C      CALCULATE CORRELATION BETWEEN SAMPLES.
          C
          C      BIG=B.
          C      IK=IJ+KBLK-1
          C      DO 68 IT=ITL,ITU
          C      SUM1=B.
          C      SUM2=B.
          C      DO 58 J=IJ,IK
          C      M=S(MODB2(J))
          C      MM=S(MODB2(J-IT))
          C      F=FLOAT(M)
          C      FF=FLOAT(MM)
          C      SUM1=SUM1+FF
          C      SUM2=SUM2+FF*FF
          C      CONTINUE
          C      IF(SUM2.LT..0001)GOTO 51
          C      B(IT)=SUM1/SUM2
          C      A(IT)=B(IT)*SUM1
          C      IF(BIG.LT.A(IT))BIG=A(IT)
          C      GOTO 68
          C
          C      51  B(IT)=B.
          C      A(IT)=B.
          C      68  CONTINUE

```

00000 IV-PLUS V02-51 15:32:32 15-APR-88 PAGE 3
 /TR:BLOCKS/VR

```

0028      C
0029      C
0030      C
0031      C
0032      C
0033      C
0034      C
0035      C
0036      C
0037      C

      FIND THE LARGEST ELEMENT OF ARRAY.

      DO 7# I=ITL,ITU
      11=I
      IF(BIG.EQ.A(I)) GOTO 88
      CONTINUE
      7#
      8#
      B(I)=AMAX1(B(I),-2.5)
      B(I)=AMIN1(B(I),2.5)
      BETA=B(I)
      RETURN
      END
  
```

```

0001 SUBROUTINE INSTRT(L1)
0002 C
0003 C
0004 C
0005 C
0006 C
0007 C
0008 C
0009 C
0010 C
0011 C
0012 C
0013 C
0014 C
0015 C
0016 C
0017 C
0018 C
0019 C
0020 C
0021 C
0022 C
0023 C
0024 C
0025 C
0026 C
0027 C
0028 C
0029 C
0030 C
0031 C
0032 C
0033 C
0034 C
0035 C
0036 C
0037 C
0038 C
0039 C
0040 C
0041 C

      PURPOSE: TO INITIALIZE THE INLOOP SYSTEM.

      V.R      DATE      NAME      COMMENTS
      0.1      16-JUL-79  J.M. KRESSE  EXISTING SOFTWARE
      1.0      17-JUL-79  J.M. KRESSE  BACKED UP
      1.3A     19-SEP-79  J.M. KRESSE  ADD BUFFER CONTROL
      1.4      26-SEP-79  J.M. KRESSE  ADD BUFFER OCCUPANCY

      SUBROUTINE INSTRT IS USED TO READ IN PARAMETER VALUES AND
      INITIALIZE QUANTITIES FOR SUBROUTINE INLOOP.

      PARAMETER MXQN=11,MXN=8,MXM=1,MXL=10,MXLN=10
      PARAMETER HINOIS=.FALSE.,LONOIS=.TRUE.
      BYTE FRAME(33),IDATE(10),ITIME(9)
      INTEGER BUFOCC
      INTEGER OLVL,QMAX,QN,OO,RANK
      LOGICAL LEVEL
      LOGICAL GAPPED
      REAL LOSNR,NOISE
      COMMON /ALPHA/ ALPHA(MXQM)
      COMMON /AN/ AN(MXQM*MXM,MXN)
      COMMON /AO/ AO(MXM,MXN)
      COMMON /B/ B(MXN)
      COMMON /BFFCAR/ BFFCAR(NRANGE)
      COMMON /BUFOCC/ BUFOCC(NRANGE)
      COMMON /CN/ CN(MXQM*MXM)
      COMMON /EASN/ EASN(MXQM*MXM)
      COMMON /EASD/ EASD(MXM)
      COMMON /F/ F(MXQM)
      COMMON /GAPPED/ GAPPED
      COMMON /INBUF/ VN(MXL)
      COMMON /LEFDIS/ LEFDIS(MXL)
      COMMON /MADAPT/ FACTOR,HISNR,LOSNR
      COMMON /MEFDIS/ MEFDIS(MXM)
      COMMON /MVARY/ LEVEL,M1,NLONO,NHINO
      COMMON /NNODEN/ NNODEN
      COMMON /NNODEO/ NNODEO
      COMMON /PARAM/ M,L,N,QMAX,MXVHN,DELTA,ALPHA,EASND,EASNM,G
      COMMON /QLVL/ OLVL(MXQM)
      COMMON /QN/ QN(MXQM*MXM,MXL)
      COMMON /QO/ QO(MXM,MXL-1)
      COMMON /RANGE/ RANGE(NRANGE+1)
      COMMON /RANK1/ RANK(MXQM*MXM)
      COMMON /SIGMN/ SIGMN(MXQM*MXM)
      COMMON /SIGMO/ SIGMO(MXM)
      COMMON /STATS/ NSAMP,VSQR,NOISE
      COMMON /T/ T(MXQM,2)
      COMMON /VHATN/ VHATN(MXQM*MXM,MXLN)
      COMMON /VHATO/ VHATO(MXM,MXLN-1)
  
```

PAGE 2

FORTRAM IV-PLUS V02-51
INSTRS.FTN /TR:BLOCKS/VR

16:55:14

02-MAY-88

C
C
C

PRINT DATE AND TIME

0042
0043
0044
0045

CALL DATE(I)DATE)
CALL TIME(I)TIME)
WRITE(6,10)IDATE,ITIME
FORMAT('DATE: ',10A1,' TIME: ',9A1)

10

```

C      READ IN FILENAME OF FILE CONTAINING PARAMETERS, INITIALIZATION
C      DATA.
C
C      TYPE " " ENTER FILENAME OF PARAMETER FILE: '
C      ACCEPT 18,NLETTTR,(FNAME(ILETTTR),ILETTTR=1,NLETTTR)
C      READ(3,28)FNAME
C      FORMAT(33A1)
C
C      INSERT END-OF-STRING MARK.
C
C      FNAME(33)=8
C      WRITE(6,38)FNAME
C      FORMAT('PARAMETER FILENAME: ',33A1)
C      OPEN (UNIT=8,NAME=FNAME,TYPE='OLD',READONLY)
C
C      READ IN SCALAR PARAMETERS
C
C      READ(8,48)I,M,N,QMAX,DELTA,ALPH,EAASND,EAASN,M,G
C      FORMAT(4I3,5G15.7)
C      MXVHN=MAX8(I-1,N)
C      WRITE(6,58)I,M,N,QMAX,MXVHN,DELTA,ALPH,EAASND,EAASN,M,G
C      FORMAT('SCALAR PARAMETERS: ',4I3,5G15.7,ALPH,EAASND,EAASN,M,G)
C      1' QMAX=',13, MXVHN=',13, DELTA=',G12.4, ALPH=',G12.4,
C      2' EAASND=',G12.4, EAASN=',G12.4, G=',G12.4)
C      L1=L
C
C      8846
C      8847
C
C      8848
C      8849
C      8850
C      8851
C
C      8852
C      8853
C      8854
C      8855
C      8856
C
C      8857

```



```

FORTRAN IV-PLUS V02-51          16:55:14      82-MAY-88          PAGE 4
INST45.FTN /TR:BLOCKS/VR

C
C
0050      READ (8,68) (F(IQ1),IQ1=1,QMAX),(ALPHA(IQ2),IQ2=1,QMAX),
0051      1*(ICOEFF,ICOEFF=1,N)
0052      FORMAT(<QMAX>F7.2/<QMAX>F7.2/<N>F7.2)
0053      WRITE(6,78)(F(IQ3),IQ3=1,QMAX),(ALPHA(IQ4),IQ4=1,QMAX),
0054      1*(ICOEFF,ICOEFF=1,N)
0055      FORMAT('BARRAY PARAMETERS:/' F:'.4X,<QMAX>F7.2/' ALPHA:'.
0056      1<QMAX>F7.2/' B= '.4X,<N>F7.2)
C
C
0060      READ IN INITIAL VALUES FOR SIGMA, COEFFICIENTS
C
0061      READ (8,88) SIGMO(1),(AO1,ICOEFF2),ICOEFF2=1,N)
0062      FORMAT(F7.2/<N>F7.2)
0063      WRITE(6,98)SIGMO(1),(AO1,ICOEFF3),ICOEFF3=1,N)
0064      FORMAT('BINITIAL CONDITIONS:/' SIGMA= '.4X,F7.2/' PREDICTOR:'.
0065      1<N>F7.2)
C
C
0070      READ IN ADAPTIVE M PARAMETERS
C
0071      READ(8,*)FACTOR,HISNR,LOSNR
0072      WRITE(6,95)FACTOR,HISNR,LOSNR
0073      FORMAT('ADAPTIVE M PARAMETERS:/'
0074      1 ' FACTOR= ',G12.4,' HISNR= ',G12.4,' LOSNR= ',G12.4)
0075      READ(8,*)T(IX3,IX4),IX4=1,2,IX3=1,QMAX)
0076      READ(8,*)RANGE
0077      READ(8,*)BFFCAR
0078      WRITE(6,*) T:'.((IX3,IX4),IX4=1,2),IX3=1,QMAX)
0079      WRITE(6,*) RANGE:'. RANGE:'. RANGE
0080      WRITE(6,*) BFFCAR:'. BFFCAR
C
C
0085      CLOSE FILE
C
0090      CLOSE (UNIT=8)

```



```

0034      1 SHAT(MODB((ITM3-T(1BLK2))),IBUF1))
0035      CALL INLOOP(V(ITM6),Q,VHAT(ITM7),BUFCNT)
0036      CALL INLOOP(V(ITM6),Q,VHAT(ITM7))
0037      CALL OUTBUF(BUFCNT,PREFIX,Q)
0038      IF(ITM5.LE.0)GO TO 10
0039      BTA=BETA(1BLK3)
0040      IT1=T(1BLK3)
0041      GO TO 20
0042      CONTINUE
0043      BTA=0.
0044      IT1=1
0045      GO TO 20
0046      CONTINUE
0047      SHAT(ITM7)=VHAT(ITM7)*ALPHA*BTA*SHAT(MODB((ITM7-IT1),IBUF1))
0048      IF(.NOT.((MOD(ITM4,KBLK).EQ.0).AND.(GAPPED.EQ.YES)))GO TO 27
0049      GAPPED=NO
0050      BETA(1BLK2)=BTASAV
0051      DO 26 IX2=1,MXOM
0052      DO 25 IX3=1,2
0053      THR(IX2,IX3)=THRSAB(IX2,IX3)
0054      CONTINUE
0055      GO TO 27
0056      CONTINUE
0057      SHAT(ITM7)=AMAX1(MINVAL,(AMIN1(MAXVAL,SHAT(ITM7))))
0058      IF(.NOT.((ITM7.EQ.IBUF1).AND.(ITM5.GT.0)))GO TO 40
0059      WRITE(LUNSH,30)(NINT(SHAT(IX1)),IX1=1,IBUF1)
0060      FORMAT(16I5)
0061      GO TO 40
0062      CONTINUE
0063      FS1=FLOAT(S(MODB(ITM5,IBUF2)))
0064      IF(BUFFIL.EQ.NO)GO TO 42
0065      WRITE(LUN8UF,41)FS1,BUFCNT
0066      FORMAT(F7.0,F10.1)
0067      GO TO 42
0068      CONTINUE
0069      ES=ES+FS1**2
0070      EN=EN+(FS1-SHAT(ITM7))**2
0071      IF(MOD(ITM5,1000).NE.0)GO TO 60
0072      WRITE(LUNPRT,50)ITM5,ES,EN
0073      FORMAT(17,2G12.4)
0074      GO TO 60
0075      CONTINUE
0076      RETURN
0077      END

```

11.3A
1-1.3A
11.2

11.6
11.6
11.6
11.6
11.6
11.6
11.6
11.6

11.2
11.2
11.2
11.2

```

FORTRAN IV-PLUS V02-51      15:59:43      15-APR-88      PAGE 1
MODB.FTH      /TR:BLOCKS/VR

0001      INTEGER FUNCTION MODB(INT1,INT2)
C
C      PURPOSE:  TO CALCULATE INT1 MOD INT2,
C      WHERE INT2=2**N, FOR SOME INTEGER N,
C      EXCEPT MODB(INT2,INT2)=INT2.
C
C      V.R      DATE      NAME      COMMENTS
C      1.1      19-JUL-79      J.M. KRESSE      ORIGINATION
C-----DECLARATIONS
C      INTEGER INT1,INT2
C-----PROCEDURE
C
0002      MODB=IAND(INT1-1,INT2-1)+1
0003      RETURN
0004      END
0005

```

11.3A
 1-1.3A

```

FORTRAN IV-PLUS V02-51      15:36:52      15-APR-80      PAGE 1
INLOO4.FTN /TR:BLOCKS/VR

0001      C
0002      C
0003      C
0004      C
0005      C
0006      C
0007      C
0008      C
0009      C
0010      C
0011      C
0012      C
0013      C
0014      C
0015      C
0016      C
0017      C

SUBROUTINE INLOOP(V,Q,VHAT,BUFCNT)
SUBROUTINE INLOOP(V,Q,VHAT)

PURPOSE: TO SIMULATE THE INLOOP SYSTEM.

V.R      DATE      NAME      COMMENTS
001      16-JUL-79      J.M. KRESSE      EXISTING SOFTWARE
1.0      17-JUL-79      J.M. KRESSE      BACKED UP
1.3A     19-SEP-79      J.M. KRESSE      ADD BUFFER CONTROL

INLOOP TAKES THE REDUCED SPEECH, V, AT TIME N, AND RETURNS THE
QUANTIZER LEVEL, Q, AND THE ESTIMATE OF THE REDUCED SPEECH,
VHAT, FOR THE TIME N-L+1 (A DELAY OF L-1 UNITS).

PARAMETER MXL=10
INTEGER QMAX,Q
REAL BUFCNT
COMMON /PARAM/M,L,N,QMAX
COMMON /INBUF/VN(MXL)
COMMON /NNODEO/NNODEO
COMMON /NNODEN/NNODEN

SAVE V IN VN

IF(L.LE.1)GO TO 10
DO 20 IBCK=1,L-1
  VN(L-IBCK+1)=VN(L-IBCK)
CONTINUE
VN(1)=V
CONTINUE

NUMBER OF NEW NODES=NUMBER OF OLD NODES*NUMBER OF QUANTIZER
LEVELS

NNODEN=NNODEO*QMAX

GROW NEW NODES

CALL GROW4(BUFCNT)
CALL GROW3

RANK NEW NODES

CALL RANKS3
  
```

11.3A
 1-1.3A

FORTAN IV-PLUS V02-51 15:36:52 15-APR-88 PAGE 2
INLOO4.FTN /TR:BLOCKS/VR

C ANNOUNCE DECISION, COLLECT STATISTICS, AND PRUNE THE TREE
C

0018 CALL PRUNE3(Q,VHAT)
0019 RETURN
0020 END

11-3A
1-1.3A

1-1.3A
1-1.3A

| SUBROUTINE GROW4(BUFCNT) | | SUBROUTINE GROW3 | | PURPOSE: TO GROW THE NEW NODES. | |
|--------------------------|-----------|------------------|--------------------|---------------------------------|--|
| V.R | DATE | NAME | COMMENTS | | |
| 0.1 | 16-JUL-79 | J.M. KRESSE | EXISTING SOFTWARE | | |
| 1.0 | 17-JUL-79 | J.M. KRESSE | BACKED UP | | |
| 1.3A | 19-SEP-79 | J.M. KRESSE | ADD BUFFER CONTROL | | |

SUBROUTINE GROW GROWS QMAX NEW NODES FROM EACH OLD NODE.

```

11.3A
PARAMETER MXQM=11,MXN=8,MXM=1,MXL=10,MXLN=10
INTEGER QMAX,QN,QO
REAL BUFCNT,BUFFAC,ENORM,T
COMMON /PARAM/M,L,N,QMAX,MXVHM,DELTA,ALPHA,EASND,EASNM,G
COMMON /F/(MXQM)/B/B(MXN)/ALPHA/ALPHA(MXQM)
COMMON /NNODE/NNODEO/VHATO/VHATO(MXM,MXLN-1)/QO/QO(MXM,MXL-1)
COMMON /EASO/EASO(MXM)
COMMON /AO/AO(MXM,MXN)/SIGMO(MXM)
COMMON /NNODEN/NNODEN/CN/CN(MXQM=MXM)/VHATN/VHATN(MXQM=MXM,MXLN)
COMMON /QN/QN(MXQM=MXM,MXL)
COMMON /EASN/EASN(MXQM=MXM)/AN/AN(MXQM=MXM,MXN)
COMMON /SIGM/SIGM(MXQM=MXM)
COMMON /INBUF/VN(MXL)
COMMON /T/T(MXQM,2)
WRITE(6,100)
FORMAT(10NEW NODES: '//3X,'Q1 2',3X,'VHAT1',7X,'VHAT2',9X,'CN',
18X,'EASN',7X,'SIGMO',8X,'AO1',9X,'AO2',8X,'SIGMN',8X,'AN1',
29X,'AN2'//)
11.3A

```

GROW QMAX NEW NODES

DO 28 IQ-1, QMAX

COMPUTE NEW NODE INDEX

$$INN = (INO - 1) * QMAX + IQ$$

COMPUTE THE ESTIMATE OF THE ERROR

$$\hat{E}HAT = F(IQ) * SIGMO(INO)$$

15:37:09

15-APR-88

PAGE 2

[illegible]

```

C
C
C
C
0043      1
          EAASN(INN)=(1.-ALPH)*ABS(VHATN(INN,1))+ALPH*
          EAASO(INO)
C
C
C
C
0044      DO 60 ICOF1=1,N
0045          CORRFA=(G*VHATN(INN,ICOF1+1)*EHAT)/
0046          (EAASN(INN)*EAASN(N)**2
0047          AN(INN,ICOF1)-DELTA*B(ICOF1)*(1.-DELTA)*
          (AO(INO,ICOF1)+CORRFA)
          CONTINUE
C
C
C
C
0048      UPDATE SIGMA
          SIGMN(INN)=AMAX1(ALPHA(IQ)*SIGMO(INO),
          (EAASN(INN)+EAASN(N))/EAASND)
          WRITE(6,110)INN,ON(INN,1),ON(INN,2),
          VHATN(INN,1),VHATN(INN,2),CH(INN),EAASN(INN),
          SIGMO(INO),AO(INO,1),AO(INO,2),SIGMN(INN),
          AN(INN,1),AN(INN,2)
          C 110
          FORMAT(13,2I2,10G12.4)
          CONTINUE
0049      20
0050      10
0051      CONTINUE
0052      RETURN
          END

```

AD-A086 133

NOTRE DAME UNIV IN DEPT OF ELECTRICAL ENGINEERING F/G 17/2
DESIGN AND IMPLEMENTATION OF A SPEECH CODING ALGORITHM AT 9600 --ETC(U)
APR 80 J L WELSA, D L COHN, A ARORA DCA100-79-C-0005

UNCLASSIFIED

ML

3 OF 3.

AD-A086 133



END
DATE
FILMED
8-80
DTIC

```

FORTRAN IV-PLUS V02-B1      15:37:35      15-APR-88      PAGE 1
BUFCON.FTN      /TR:BLOCKS/VR

00001      SUBROUTINE BUFCON(BUFCNT,BUFFAC)
C
C      PURPOSE:  TO COMPUTE THE BUFFER CONTROL FACTOR.
C
C      V.R      DATE      NAME      COMMENTS
C      1.3A    19-SEP-79    J.M. KRESSE    ADD BUFFER CONTROL
C      1.4      26-SEP-79    J.M. KRESSE    ADD BUFFER OCCUPANCY
C
C-----DECLARATIONS
C
00002      PARAMETER NRange=15
00003      INTEGER BUFOCC
00004      INTEGER IX
00005      REAL BFFCAR,BUFCNT,BUFFAC,RANGE
00006      COMMON /RANGE/ RANGE(NRange+1)
00007      COMMON /BFFCAR/ BFFCAR(NRange)
00008      COMMON /BUFOCC/ BUFOCC(NRange)
C
C-----PROCEDURE
C
00009      DO 15 IX=1,NRange
00010      IF(.NOT.((RANGE(IX).LE.BUFCNT).AND.(BUFCNT.LT.
1      RANGE(IX+1))))GO TO 5
00011      BUFFAC=BFFCAR(IX)
00012      BUFOCC(IX)=BUFOCC(IX)+1
00013      GO TO 5
00014      CONTINUE
C      IF((RANGE(IX).LE.BUFCNT).AND.(BUFCNT.LE.RANGE(IX+1)))
1      BUFFAC=BFFCAR(IX)
00015      CONTINUE
00016      RETURN
00017      END

```

11.4

11.4

11.4
11.4
11.4
11.4
11.4
11.4
11.4
11.4

```

FORTRAN IV-PLUS V82-51      15:37:46      15-APR-88      PAGE 1
RANKS4.FTN  /TR:BLOCKS/VR

SUBROUTINE RANKS3
C
C      PURPOSE: TO RANK THE NEW NODES.
C
C      V.R      DATE      NAME      COMMENTS
C      8.1      16-JUL-79      J.M. KRESSE      EXISTING SOFTWARE
C      1.8      17-JUL-79      J.M. KRESSE      BACKED UP
C
C      SUBROUTINE RANKS RANKS THE NEW NODES IN DESCENDING ORDER OF THE
C      CRITERION. THE ARRAY RANK STORES NODE INDICES: E.G., RANK(1)=
C      NODE INDEX OF NEW NODE WITH HIGHEST VALUE OF CRITERION.
C
C      PARAMETER MXQM=11,MXM=1
C      INTEGER RANK,TEMP
C      COMMON /NNODEN/NNODEN/CN/CN(MXQM=MXM)
C      COMMON /RANK1/RANK(MXQM=MXM)
C
C      INITIALIZE RANKINGS
C
C      DO 18 NINDX=1,NNODEN
C
C          INITIAL RANK IS NODE INDEX
C          RANK(NINDX)=NINDX
C
C      CONTINUE
18

```

```

FORTRAN IV-PLUS V02-51      15:37:46      15-APR-88      PAGE 2
RANKS4.FTN      /TR:BLOCKS/VR

      C
      C      PERFORM BUBBLE SORT
      C
      C      DO (NNODEN-1) PASSES
      C
      C      DO 28 NPASS=1,NNODEN-1
0009
      C
      C      DO (NNODEN-NPASS) COMPARISONS ON THE PASS
      C
      C      DO 38 NCOMP=1,NNODEN-NPASS
0010
      C
      C      THE COMPARISON:
      C
      C      IF(CN(RANK(NCOMP)),GE,CN(RANK(NCOMP+1)))GO TO 48
      C
      C      ORDER REVERSED: REVERSE THEM
      C
      C      TEMP=RANK(NCOMP)
      C      RANK(NCOMP)=RANK(NCOMP+1)
      C      RANK(NCOMP+1)=TEMP
      C
      C      CONTINUE
      C
      C      CONTINUE
      C      WRITE(6,58)(RANK(NINDX1),NINDX1=1,NNODEN)
      C      FORMAT(/,58THE RANKING: '//<NNODEN>I3//)
      C      RETURN
      C      END
0018
0019
48
38
28
C 58

```


PAGE 2

15-APR-88

18:38:55

FORTRAN IV-PLUS V82-B1
PRUNES.FTN /TR:BLOCKS/VR

C

OLVL(Q)=OLVL(Q)+1

2932


```

FORTRAN IV-PLUS V02-51      15:38:55      15-APR-88      PAGE 3
PRUNES.FTN /TR:BLOCKS/VR

      C
      C      COLLECT EFFECTIVE L STATISTICS
0033      C      CALL LEFST
      C
      C      COLLECT EFFECTIVE M STATISTICS
0034      C      MEFDIS(NNODEO)=MEFDIS(NNODEO)+1
0035      C      IF(OPLOT.EQ.'Y')CALL MFILE
      C
      C      PRUNE THE TREE BY NOT SAVING INFORMATION FROM THE PRUNED NODES.
      C
      C      IF(LEVEL.EQV.HINOIS)GO TO 28
      C      NLONO=NLONO+1
      C      GO TO 38
      C      28      NHINO=NHINO+1
      C      GO TO 38
      C      38      CONTINUE
      C      CALL SNRVAL(-CN(RANK(1)),SNR)
      C      IF(.NOT.((LEVEL.EQV.LONOIS).AND.(SNR.LT.LOSNR)))GO TO 48
      C      LEVEL=HINOIS
      C      M=M1
      C      GO TO 48
      C      48      CONTINUE
      C      IF(.NOT.((LEVEL.EQV.HINOIS).AND.(SNR.GT.HISNR)))GO TO 58
      C      LEVEL=LONOIS
      C      M=1
      C      GO TO 58
      C      58      CONTINUE
      C      NNODEO=N
      C      DO 78 INN=1,NNODEN
      C      IF(.NOT.
      C      1      ((QN(RANK(INN),L).EQ.QN(RANK(1),L))
      C      2      .AND.(NNODEO.LT.M)
      C      3      .AND.((-CN(RANK(INN))).LE.FACTOR*(-CN(RANK(1))))))
      C      4      GO TO 68
      C      NNODEO=NNODEO+1
      C      CALL SAVE(INN)
      C      CONTINUE
      C      68      CONTINUE
      C      RETURN
      C      78      END
0056
0057
0058
0059
0060
0061

```

```

FORTRAN IV-PLUS V02-51      15:38:26      15-APR-88      PAGE 1
LEFT.FTN /TR:BLOCKS/VR

0001      SUBROUTINE LEFTST
0002      PURPOSE: TO COLLECT THE EFFECTIVE L STATISTICS.
0003      V.R      DATE      NAME      COMMENTS
0004      0.1      16-JUL-79      J.M. KRESSE      EXISTING SOFTWARE
0005      1.0      17-JUL-79      J.M. KRESSE      BACKED UP
0006      LEFTST COLLECTS EFFECTIVE L STATISTICS
0007      PARAMETER MXL=10,MYM=1,MYOM=11
0008      PARAMETER DIFF=.FALSE.,SAME=.TRUE.
0009      INTEGER ON
0010      LOGICAL COND
0011      COMMON /LEFDIS/ LEFDIS(MXL)
0012      COMMON /NNODEN/ NNODEN
0013      COMMON /PARAM/ M,L
0014      COMMON /QN/ QN(MYOM*MYM,MXL)
0015      COND=SAME
0016      LEVEL=L
0017      DO UNTIL (DIFFERENCE FOUND)
0018      CONTINUE
0019      NODE=2
0020      DO UNTIL ((OUT OF NODES) OR (DIFFERENCE FOUND))
0021      CONTINUE
0022      IF(QN(NODE,LEVEL).NE.QN(1,LEVEL))COND=DIFF
0023      IF((NODE.EQ.NNODEN).OR.(COND.EQV.DIFF))GO TO 30
0024      NODE=NODE+1
0025      GO TO 20
0026      CONTINUE
0027      IF(COND.EQV.DIFF)GO TO 40
0028      LEVEL=LEVEL+1
0029      GO TO 10
0030      CONTINUE
0031      LEFDIS(LEVEL)=LEFDIS(LEVEL)+1
0032      RETURN
0033      END

```

```

FORTRAN IV-PLUS V02-51      15:38:38      15-APR-88      PAGE 1
MFILE.FTN /TR:BLOCKS/VR

0001      SUBROUTINE MFILE
C
C      PURPOSE:  TO COLLECT THE EFFECTIVE M STATISTICS AND PUT THEM OUT
C               ON LOGICAL UNIT 4.
C
C      V.R      DATE      NAME      COMMENTS
C      0.1      16-JUL-79      J.M. KRESSE      EXISTING SOFTWARE
C      1.0      17-JUL-79      J.M. KRESSE      BACKED UP
C-----DECLARATIONS
C      PARAMETER BUFFL=16
C      COMMON /ICNT/ ICNT
C      COMMON /MBUFF/ MBUFF/ MBUFF(BUFFL)
C      COMMON /NNODEO/ NNODEO
C-----PROCEDURE
C
C      ICNT=ICNT+1
C      MBUFF(ICNT)=NNODEO
C      IF(ICNT.NE.BUFFL)GO TO 20
C      ICNT=0
C      WRITE(4,10)MBUFF
C      FORMAT(<BUFFL>I5)
C      CONTINUE
C      RETURN
C      END
10
20
0002
0003
0004
0005
0006
0007
0008
0009
0010
0011
0012
0013
0014

```

```

FORTRAN IV-PLUS V02-51          15:38:49    15-APR-80    PAGE 1
SNRVAL.FTN      /TR:BLOCKS/VR

0001      SUBROUTINE SNRVAL( NOISE, SNR )
C
C      PURPOSE:  TO CALCULATE THE SNR CORRESPONDING TO NOISE.
C
C      V.R      DATE      NAME      COMMENTS
C      0.1      16-JUL-79    J.M. KRESSE  EXISTING SOFTWARE
C      1.0      17-JUL-79    J.M. KRESSE  BACKED UP
C
C-----DECLARATIONS
C
0002      PARAMETER MXL=10
0003      PARAMETER MAXVAL=1.E30, MINVAL=-1.E30
0004      REAL NOISE
0005      COMMON /INBUF/ VN(MXL)
0006      COMMON /PARAM/ M,L
C
C-----PROCEDURE
C
0007      IF( NOISE.NE.0. ) GO TO 10
0008      GO TO 50
0009      SNR=MAXVAL
0010      CONTINUE
0011      SIG=0.
0012      DO 20 IX=1,L
0013          SIG=SIG+VN(IX)**2
0014      CONTINUE
0015      IF( SIG.NE.0. ) GO TO 30
0016      SNR=MINVAL
0017      GO TO 40
0018      CONTINUE
0019      SNR=10.*ALOG10( SIG/NOISE )
0020      SNR=AMAX1( SNR, MINVAL )
0021      SNR=AMIN1( SNR, MAXVAL )
0022      GO TO 40
0023      CONTINUE
0024      GO TO 50
0025      CONTINUE
0026      RETURN
0027      END

```

```

SUBROUTINE OUTBUF(BUFCNT,PREFIX,Q)
C
C PURPOSE: TO COMPUTE THE SIZE OF THE BUFFER AFTER EACH SAMPLE.
C
C V.R DATE NAME COMMENTS
C 1.2 22-JUL-79 J.M. KRESSE ORIGINATION
C 1.3A 19-SEP-79 J.M. KRESSE ADD BUFFER CONTROL
C 1.4 08-OCT-79 J.M. KRESSE NEW SOURCE CODE
C
C-----DECLARATIONS
C
C PARAMETER RUNLEN=14,RNCOLN=4.
C PARAMETER R2NLEN=10,R2COLN=7.
C PARAMETER R2NLEN=14,R2COLN=4.
C PARAMETER QMAX=11,XMTCNT=1.35
C PARAMETER QMAX=11,XMTCNT=1.5
C INTEGER PREFIX,Q
C INTEGER R2CNT
C REAL BITCNT(QMAX),BUFCNT
C REAL BITCNT(QMAX,0:1),BUFCNT
C
C-----DEFINITIONS
C
C DATA BITCNT /0.,3.,3.,4.,4.,5.,6.,7.,7.,7.,7.,7./
C DATA BITCNT /5.3.,3.,6.,6.,10.,8.,10.,11.,11.,10.,5./
C 1 5.2.5.3.5.6.5.6.5.9.5.7.5.10.5.9.5.10.5.10.5/
C
C-----PROCEDURE
C
C IF(Q.EQ.1)GO TO 10
C BUFCNT=BUFCNT+PREFIX+BITCNT(Q)
C R2CNT=PREFIX/R2NLEN
C BUFCNT=BUFCNT+FLOAT(R2CNT)*R2COLN+FLOAT(PREFIX-
C 1 R2CNT*R2NLEN)+BITCNT(Q)
C PREFIX=0
C GO TO 30
C CONTINUE
C
C PREFIX=PREFIX+1
C IF(PREFIX.NE.RUNLEN)GO TO 20
C BUFCNT=BUFCNT+RNCOLN
C PREFIX=0
C GO TO 20
C CONTINUE
C
C GO TO 30
C CONTINUE
C
C BUFCNT=AMAX1(0.,BUFCNT-XMTCNT)
C BUFCNT=AMAX1(0.,BUFCNT+BITCNT(Q,PREFIX)-XMTCNT)
C BUFCNT=BUFCNT+BITCNT(Q,PREFIX)-XMTCNT
C IF(.NOT.((PREFIX.EQ.0).AND.(Q.EQ.1)))GO TO 10
C PREFIX=1
C GO TO 20
C CONTINUE
C 10
C CONTINUE
C 20
C CONTINUE
C 30
C CONTINUE
C
C 10
C CONTINUE
C 20
C CONTINUE
C 30
C CONTINUE
C 40
C CONTINUE
C 50
C CONTINUE
C 60
C CONTINUE
C 70
C CONTINUE
C 80
C CONTINUE
C 90
C CONTINUE
C 100
C CONTINUE
C 110
C CONTINUE
C 120
C CONTINUE
C 130
C CONTINUE
C 140
C CONTINUE
C 150
C CONTINUE
C 160
C CONTINUE
C 170
C CONTINUE
C 180
C CONTINUE
C 190
C CONTINUE
C 200
C CONTINUE
C 210
C CONTINUE
C 220
C CONTINUE
C 230
C CONTINUE
C 240
C CONTINUE
C 250
C CONTINUE
C 260
C CONTINUE
C 270
C CONTINUE
C 280
C CONTINUE
C 290
C CONTINUE
C 300
C CONTINUE
C 310
C CONTINUE
C 320
C CONTINUE
C 330
C CONTINUE
C 340
C CONTINUE
C 350
C CONTINUE
C 360
C CONTINUE
C 370
C CONTINUE
C 380
C CONTINUE
C 390
C CONTINUE
C 400
C CONTINUE
C 410
C CONTINUE
C 420
C CONTINUE
C 430
C CONTINUE
C 440
C CONTINUE
C 450
C CONTINUE
C 460
C CONTINUE
C 470
C CONTINUE
C 480
C CONTINUE
C 490
C CONTINUE
C 500
C CONTINUE
C 510
C CONTINUE
C 520
C CONTINUE
C 530
C CONTINUE
C 540
C CONTINUE
C 550
C CONTINUE
C 560
C CONTINUE
C 570
C CONTINUE
C 580
C CONTINUE
C 590
C CONTINUE
C 600
C CONTINUE
C 610
C CONTINUE
C 620
C CONTINUE
C 630
C CONTINUE
C 640
C CONTINUE
C 650
C CONTINUE
C 660
C CONTINUE
C 670
C CONTINUE
C 680
C CONTINUE
C 690
C CONTINUE
C 700
C CONTINUE
C 710
C CONTINUE
C 720
C CONTINUE
C 730
C CONTINUE
C 740
C CONTINUE
C 750
C CONTINUE
C 760
C CONTINUE
C 770
C CONTINUE
C 780
C CONTINUE
C 790
C CONTINUE
C 800
C CONTINUE
C 810
C CONTINUE
C 820
C CONTINUE
C 830
C CONTINUE
C 840
C CONTINUE
C 850
C CONTINUE
C 860
C CONTINUE
C 870
C CONTINUE
C 880
C CONTINUE
C 890
C CONTINUE
C 900
C CONTINUE
C 910
C CONTINUE
C 920
C CONTINUE
C 930
C CONTINUE
C 940
C CONTINUE
C 950
C CONTINUE
C 960
C CONTINUE
C 970
C CONTINUE
C 980
C CONTINUE
C 990
C CONTINUE
C 1000
C CONTINUE

```

FORTRAN IV-PLUS V02-S1
OUTBUF.PTH /TR:BLOCKS/VR
0020 END

15:39:21

15-APR-88

PAGE 2

FORTRAN IV-PLUS V02-51 15:39:35 15-APR-98

PAGE 1

TR: KS/

```

0001 SUBROUTINE INEND
C
C PURPOSE: TO COLLECT AND OUTPUT STATISTICAL DATA.
C
C V.R DATE NAME COMMENTS
C 0.1 16-JUL-79 J.M. KRESSE EXISTING SOFTWARE
C 1.0 17-JUL-79 J.M. KRESSE BACKED UP
C 1.4 26-SEP-79 J.M. KRESSE ADD BUFFER OCCUPANCY
C
C SUBROUTINE INEND GATHERS AND OUTPUTS STATISTICAL DATA
C
C PARAMETER MXL=10,MXM=1,MXOM=11
C PARAMETER N RANGE=10
C INTEGER BUFOCC
C INTEGER QLVL,QMAX
C LOGICAL LEVEL
C REAL NOISE
C COMMON /BUFOCC/ BUFOCC(N RANGE)
C COMMON /LEFDIS/ LEFDIS(MXL)
C COMMON /MEFDIS/ MEFDIS(MXM)
C COMMON /MVARY/ LEVEL,M1,NLONO,NHINO
C COMMON /PARAM/ M.L,N,QMAX
C COMMON /QLVL/ QLVL(MXOM)
C COMMON /STATS/ NSAMP,VSQR,NOISE
C
C OUTPUT THE NUMBER OF SAMPLES HANDLED BY THE TREE SEARCHING
C ALGORITHM.
C
C WRITE(6,10)NSAMP
C FORMAT('0 OF SAMPLES (TREE SEARCHING)= ',15)
C WRITE(6,20)NLONO,NHINO
C FORMAT('0 OF SAMPLES (M=1)= ',15,'0 OF SAMPLES (M=M)= ',15)
C
C OUTPUT SNR FOR TREE SEARCHING
C
C WRITE(6,30)10,'ALOG10(VSQR/NOISE)'
C FORMAT(' SNR FOR TREE SEARCHING= ',G12.4,' DB ')

```

11.4
11.4

11.4

FORTRAN IV-PLUS V02-51 15:39:35 15-APR-80 PAGE 2
INEND5.FTN /TR:BLOCKS/VR

```

C
C
C      OUTPUT QUANTIZER LEVEL FREQUENCY DISTRIBUTION AND ENTROPY
C
      ALOG2=ALOG(2.)
      ENTRPY=0.
      WRITE(6,40)
      FORMAT('QUANTIZER LEVEL FREQUENCY DISTRIBUTION: '//
1' LEVEL NUMBER',5X,'P',7X,'-P=LOG(P)')
      DO 80 IQ1=1,OMAX
          FREQQ=FLOAT(OLVL(IQ1))/FLOAT(NSAMP)
          IF(FREQQ.EQ.0.)GO TO 50
          ANLN=-FREQQ*ALOG(FREQQ)/ALOG2
          GO TO 60
          ANLN=0.
          GO TO 60
          CONTINUE
          ENTRPY=ENTRPY+ANLN
          WRITE(6,70)IQ1,OLVL(IQ1),FREQQ,ANLN
          FORMAT(14,4X,15,2G12.4)
          CONTINUE
          WRITE(6,90)ENTRPY
          FORMAT('CORRESPONDING ENTROPY: ',G12.4,' BITS/SAMPLE ')
          WRITE(6,100)
          FORMAT('BL(EFF)',3X,'X',5X,'0')
          WRITE(6,110)(IX1,(1000.-=FLOAT(LEFDIS(IX1)))/FLOAT(NSAMP),
1 LEFDIS(IX1),IX1=1,L)
          FORMAT(15,F8.2,16)
          WRITE(6,120)
          FORMAT('OM(EFF)',3X,'X',5X,'0')
          WRITE(6,130)(IX2,(1000.-=FLOAT(MEFDIS(IX2)))/FLOAT(NSAMP),
1 MEFDIS(IX2),IX2=1,M1)
          FORMAT(15,F8.2,16)
          WRITE(6,*)'BUFFER OCCUPANCY:',BUFOCC
          RETURN
          END

```

11.4

APPENDIX B

SEGMENTED SNR PLOTS

B.1. Introduction

A number of similar programs were developed to aid in the analysis of the performance of PARC. The purpose of the programs were to generate plots, called SNRSIG, which indicate graphically the short-time performance of the system versus the short-time signal level. This information proved useful, as it appears that the average performance of the systems over short periods of time is more indicative than the overall average performance.

There are three groups of programs, with two programs in each group. The first program in each group performs the short-time analysis, and generates the data to be plotted. The second program then takes the data and generates the actual plot.

The first group, DBCALC and SNRSIG, deals with the short-time signal-to-noise ratio. This group is useful in analyzing the performance of the quantizer, especially problems like slope overload noise and granular noise. The second group, BFCALC and BFPLOT, deals with the average buffer length, and the third group, BFDIFF and BFDFPL, deals with the difference in the average buffer length. These programs are useful in analyzing the performance of the source coding and of the buffer control.


```

FORTRAN IV-PLUS V02-51      14:36:54      15-APR-80      PAGE 1
DBCALC.FTN /TR:BLOCKS/VR

0001      PROGRAM DBCALC
C
C      PURPOSE:  TO CALCULATE THE LOCAL SIGNAL STRENGTH AND THE LOCAL
C               SNR FOR A SPEECH FILE AND AN SHAT FILE.
C
C      V.R      DATE      NAME      COMMENTS
C      0.1      29-JUN-79      J.M. KRESSE      ORIGINATION
C      0.2      12-JUL-79      J.M. KRESSE      TO ADD THE SNR HEADER
C
C      -----DECLARATIONS
C
0002      PARAMETER DIMENS=33,MAXBUF=16,MAXLEN=1000,MAXVAL=1.E30
0003      PARAMETER MINVAL=-1.E30,NO=.FALSE.,YES=.TRUE.
0004      BYTE DBFILE(DIMENS),SFILE(DIMENS),SHFILE(DIMENS)
0005      INTEGER BLKLEN,SPOINT,BUFLEN,LUNDB,LUNSH,LUNSCR,LUNSH
0006      INTEGER S(MAXLEN),SBUF(MAXBUF),SHAT(MAXLEN),SHBUF(MAXBUF),SPOINT
0007      LOGICAL EOF,MORES
0008      REAL OVRSNR,SIGSTR,SNR,TOTSIG,TOTNOI
C
C      -----INITIALIZATION
C
0009      BPOINT=MAXBUF
0010      BUFLN=MAXBUF
0011      EOF=NO
0012      MORES=.YES
0013      TOTNOI=.0
0014      TOTSIG=.0
C
C      -----PROCEDURE
C
0015      ASK FOR FILENAMES
C
0016      CALL ASK1(SFILE,SHFILE,DBFILE,DIMENS,LUNS,LUNSH,LUNDB,LUNSCR)
C
0017      OPEN AND READY THE FILES
C
0018      CALL OPENER(SFILE,SHFILE,DBFILE,DIMENS,LUNS,LUNSH,LUNDB,LUNSCR)
C
0019      ASK FOR THE DESIRED CALCULATION BLOCK LENGTH.
C
0020      CALL ASK2(MAXLEN,BLKLEN)
C
0021      DO UNTIL (MORES.EQ.NO)
C
0022      CONTINUE
C
0023      GET S. SHAT
C
0024      CALL GETSEM(S,SHAT,SBUF,SHBUF,LUNS,LUNSH,BLKLEN,SPOINT,
0025      MAXBUF,BUFLN,BPOINT,EOF,MORES)
C
0026      IF (SPOINT.NE.0)
0027      IF (SPOINT.EQ.0)GO TO 20
C
0028      CALCULATE THE LOCAL SIGNAL STRENGTH AND SNR.

```



```

FORTRAN IV-PLUS V82-51      14:37:14      15-APR-88      PAGE 1
ASK1.FTN      /TR:BLOCKS/VR
0001      SUBROUTINE ASK1(SFILE,SHFILE,DBFILE,DIMENS)
C
C      PURPOSE: TO ASK FOR THE FILENAMES NEEDED
C
C      V.R      NAME      DATE      COMMENTS
C      0.1      J.M. KRESSE      26-JUN-79      ORIGINATION
C
C-----DECLARATIONS
C
C      PARAMETER NUL=' '
C      INTEGER DIMENS
C      BYTE DBFILE(DIMENS),SFILE(DIMENS),SHFILE(DIMENS)
C
C-----PROCEDURE
C
0005      WRITE(5,10)
0006      FORMAT(' WHAT IS THE SPEECH FILENAME? ')
0007      READ(5,20)SFILE
0008      FORMAT(<DIMENS>A1)
0009      WRITE(5,30)
0010      FORMAT(' WHAT IS THE SHAT FILENAME? ')
0011      READ(5,40)SHFILE
0012      FORMAT(<DIMENS>A1)
0013      WRITE(5,50)
0014      FORMAT(' WHAT DO YOU WANT TO CALL THE OUTPUT FILE? ')
0015      READ(5,60)DBFILE
0016      FORMAT(<DIMENS>A1)
C
C      SET END-OF-STRING MARKS
C
C      SFILE(DIMENS)=NUL
C      SHFILE(DIMENS)=NUL
C      DBFILE(DIMENS)=NUL
C
C      RETURN
C      END
0020
0021

```

```

0001 SUBROUTINE OPENER(SF,SHFILE,DBFILE,DIMENS,
0002 1 LUNS,LUNSH,LUNDB,LUNSCR)
0003
0004 PURPOSE: TO OPEN AND READY THE FILES REQUESTED.
0005
0006 V.R DATE NAME COMMENTS
0007 0.1 26-JUN-79 J.M. KRESSE ORIGINATION
0008 0.2 12-JUL-79 J.M. KRESSE TO ADD THE SNR HEADER
0009
0010 C-----DECLARATIONS
0011 C
0012 C
0013 C
0014 C
0015 C
0016 C
0017 C
0018 C
0019 C
0020 C
0021 C
0022 C
0023 C
0024 C
0025 C
0026 C
0027 C
0028 C
0029 C
0030 C
0031 C
0032 C
0033 C
0034 C
0035 C
0036 C
0037 C
0038 C
0039 C
0040 C
0041 C
0042 C
0043 C
0044 C
0045 C
0046 C
0047 C
0048 C
0049 C
0050 C
0051 C
0052 C
0053 C
0054 C
0055 C
0056 C
0057 C
0058 C
0059 C
0060 C
0061 C
0062 C
0063 C
0064 C
0065 C
0066 C
0067 C
0068 C
0069 C
0070 C
0071 C
0072 C
0073 C
0074 C
0075 C
0076 C
0077 C
0078 C
0079 C
0080 C
0081 C
0082 C
0083 C
0084 C
0085 C
0086 C
0087 C
0088 C
0089 C
0090 C
0091 C
0092 C
0093 C
0094 C
0095 C
0096 C
0097 C
0098 C
0099 C
0100 C
0101 C
0102 C
0103 C
0104 C
0105 C
0106 C
0107 C
0108 C
0109 C
0110 C
0111 C
0112 C
0113 C
0114 C
0115 C
0116 C
0117 C
0118 C
0119 C
0120 C
0121 C
0122 C
0123 C
0124 C
0125 C
0126 C
0127 C
0128 C
0129 C
0130 C
0131 C
0132 C
0133 C
0134 C
0135 C
0136 C
0137 C
0138 C
0139 C
0140 C
0141 C
0142 C
0143 C
0144 C
0145 C
0146 C
0147 C
0148 C
0149 C
0150 C
0151 C
0152 C
0153 C
0154 C
0155 C
0156 C
0157 C
0158 C
0159 C
0160 C
0161 C
0162 C
0163 C
0164 C
0165 C
0166 C
0167 C
0168 C
0169 C
0170 C
0171 C
0172 C
0173 C
0174 C
0175 C
0176 C
0177 C
0178 C
0179 C
0180 C
0181 C
0182 C
0183 C
0184 C
0185 C
0186 C
0187 C
0188 C
0189 C
0190 C
0191 C
0192 C
0193 C
0194 C
0195 C
0196 C
0197 C
0198 C
0199 C
0200 C
0201 C
0202 C
0203 C
0204 C
0205 C
0206 C
0207 C
0208 C
0209 C
0210 C
0211 C
0212 C
0213 C
0214 C
0215 C
0216 C
0217 C
0218 C
0219 C
0220 C
0221 C
0222 C
0223 C
0224 C
0225 C
0226 C
0227 C
0228 C
0229 C
0230 C
0231 C
0232 C
0233 C
0234 C
0235 C
0236 C
0237 C
0238 C
0239 C
0240 C
0241 C
0242 C
0243 C
0244 C
0245 C
0246 C
0247 C
0248 C
0249 C
0250 C
0251 C
0252 C
0253 C
0254 C
0255 C
0256 C
0257 C
0258 C
0259 C
0260 C
0261 C
0262 C
0263 C
0264 C
0265 C
0266 C
0267 C
0268 C
0269 C
0270 C
0271 C
0272 C
0273 C
0274 C
0275 C
0276 C
0277 C
0278 C
0279 C
0280 C
0281 C
0282 C
0283 C
0284 C
0285 C
0286 C
0287 C
0288 C
0289 C
0290 C
0291 C
0292 C
0293 C
0294 C
0295 C
0296 C
0297 C
0298 C
0299 C
0300 C
0301 C
0302 C
0303 C
0304 C
0305 C
0306 C
0307 C
0308 C
0309 C
0310 C
0311 C
0312 C
0313 C
0314 C
0315 C
0316 C
0317 C
0318 C
0319 C
0320 C
0321 C
0322 C
0323 C
0324 C
0325 C
0326 C
0327 C
0328 C
0329 C
0330 C
0331 C
0332 C
0333 C
0334 C
0335 C
0336 C
0337 C
0338 C
0339 C
0340 C
0341 C
0342 C
0343 C
0344 C
0345 C
0346 C
0347 C
0348 C
0349 C
0350 C
0351 C
0352 C
0353 C
0354 C
0355 C
0356 C
0357 C
0358 C
0359 C
0360 C
0361 C
0362 C
0363 C
0364 C
0365 C
0366 C
0367 C
0368 C
0369 C
0370 C
0371 C
0372 C
0373 C
0374 C
0375 C
0376 C
0377 C
0378 C
0379 C
0380 C
0381 C
0382 C
0383 C
0384 C
0385 C
0386 C
0387 C
0388 C
0389 C
0390 C
0391 C
0392 C
0393 C
0394 C
0395 C
0396 C
0397 C
0398 C
0399 C
0400 C
0401 C
0402 C
0403 C
0404 C
0405 C
0406 C
0407 C
0408 C
0409 C
0410 C
0411 C
0412 C
0413 C
0414 C
0415 C
0416 C
0417 C
0418 C
0419 C
0420 C
0421 C
0422 C
0423 C
0424 C
0425 C
0426 C
0427 C
0428 C
0429 C
0430 C
0431 C
0432 C
0433 C
0434 C
0435 C
0436 C
0437 C
0438 C
0439 C
0440 C
0441 C
0442 C
0443 C
0444 C
0445 C
0446 C
0447 C
0448 C
0449 C
0450 C
0451 C
0452 C
0453 C
0454 C
0455 C
0456 C
0457 C
0458 C
0459 C
0460 C
0461 C
0462 C
0463 C
0464 C
0465 C
0466 C
0467 C
0468 C
0469 C
0470 C
0471 C
0472 C
0473 C
0474 C
0475 C
0476 C
0477 C
0478 C
0479 C
0480 C
0481 C
0482 C
0483 C
0484 C
0485 C
0486 C
0487 C
0488 C
0489 C
0490 C
0491 C
0492 C
0493 C
0494 C
0495 C
0496 C
0497 C
0498 C
0499 C
0500 C
0501 C
0502 C
0503 C
0504 C
0505 C
0506 C
0507 C
0508 C
0509 C
0510 C
0511 C
0512 C
0513 C
0514 C
0515 C
0516 C
0517 C
0518 C
0519 C
0520 C
0521 C
0522 C
0523 C
0524 C
0525 C
0526 C
0527 C
0528 C
0529 C
0530 C
0531 C
0532 C
0533 C
0534 C
0535 C
0536 C
0537 C
0538 C
0539 C
0540 C
0541 C
0542 C
0543 C
0544 C
0545 C
0546 C
0547 C
0548 C
0549 C
0550 C
0551 C
0552 C
0553 C
0554 C
0555 C
0556 C
0557 C
0558 C
0559 C
0560 C
0561 C
0562 C
0563 C
0564 C
0565 C
0566 C
0567 C
0568 C
0569 C
0570 C
0571 C
0572 C
0573 C
0574 C
0575 C
0576 C
0577 C
0578 C
0579 C
0580 C
0581 C
0582 C
0583 C
0584 C
0585 C
0586 C
0587 C
0588 C
0589 C
0590 C
0591 C
0592 C
0593 C
0594 C
0595 C
0596 C
0597 C
0598 C
0599 C
0600 C
0601 C
0602 C
0603 C
0604 C
0605 C
0606 C
0607 C
0608 C
0609 C
0610 C
0611 C
0612 C
0613 C
0614 C
0615 C
0616 C
0617 C
0618 C
0619 C
0620 C
0621 C
0622 C
0623 C
0624 C
0625 C
0626 C
0627 C
0628 C
0629 C
0630 C
0631 C
0632 C
0633 C
0634 C
0635 C
0636 C
0637 C
0638 C
0639 C
0640 C
0641 C
0642 C
0643 C
0644 C
0645 C
0646 C
0647 C
0648 C
0649 C
0650 C
0651 C
0652 C
0653 C
0654 C
0655 C
0656 C
0657 C
0658 C
0659 C
0660 C
0661 C
0662 C
0663 C
0664 C
0665 C
0666 C
0667 C
0668 C
0669 C
0670 C
0671 C
0672 C
0673 C
0674 C
0675 C
0676 C
0677 C
0678 C
0679 C
0680 C
0681 C
0682 C
0683 C
0684 C
0685 C
0686 C
0687 C
0688 C
0689 C
0690 C
0691 C
0692 C
0693 C
0694 C
0695 C
0696 C
0697 C
0698 C
0699 C
0700 C
0701 C
0702 C
0703 C
0704 C
0705 C
0706 C
0707 C
0708 C
0709 C
0710 C
0711 C
0712 C
0713 C
0714 C
0715 C
0716 C
0717 C
0718 C
0719 C
0720 C
0721 C
0722 C
0723 C
0724 C
0725 C
0726 C
0727 C
0728 C
0729 C
0730 C
0731 C
0732 C
0733 C
0734 C
0735 C
0736 C
0737 C
0738 C
0739 C
0740 C
0741 C
0742 C
0743 C
0744 C
0745 C
0746 C
0747 C
0748 C
0749 C
0750 C
0751 C
0752 C
0753 C
0754 C
0755 C
0756 C
0757 C
0758 C
0759 C
0760 C
0761 C
0762 C
0763 C
0764 C
0765 C
0766 C
0767 C
0768 C
0769 C
0770 C
0771 C
0772 C
0773 C
0774 C
0775 C
0776 C
0777 C
0778 C
0779 C
0780 C
0781 C
0782 C
0783 C
0784 C
0785 C
0786 C
0787 C
0788 C
0789 C
0790 C
0791 C
0792 C
0793 C
0794 C
0795 C
0796 C
0797 C
0798 C
0799 C
0800 C
0801 C
0802 C
0803 C
0804 C
0805 C
0806 C
0807 C
0808 C
0809 C
0810 C
0811 C
0812 C
0813 C
0814 C
0815 C
0816 C
0817 C
0818 C
0819 C
0820 C
0821 C
0822 C
0823 C
0824 C
0825 C
0826 C
0827 C
0828 C
0829 C
0830 C
0831 C
0832 C
0833 C
0834 C
0835 C
0836 C
0837 C
0838 C
0839 C
0840 C
0841 C
0842 C
0843 C
0844 C
0845 C
0846 C
0847 C
0848 C
0849 C
0850 C
0851 C
0852 C
0853 C
0854 C
0855 C
0856 C
0857 C
0858 C
0859 C
0860 C
0861 C
0862 C
0863 C
0864 C
0865 C
0866 C
0867 C
0868 C
0869 C
0870 C
0871 C
0872 C
0873 C
0874 C
0875 C
0876 C
0877 C
0878 C
0879 C
0880 C
0881 C
0882 C
0883 C
0884 C
0885 C
0886 C
0887 C
0888 C
0889 C
0890 C
0891 C
0892 C
0893 C
0894 C
0895 C
0896 C
0897 C
0898 C
0899 C
0900 C
0901 C
0902 C
0903 C
0904 C
0905 C
0906 C
0907 C
0908 C
0909 C
0910 C
0911 C
0912 C
0913 C
0914 C
0915 C
0916 C
0917 C
0918 C
0919 C
0920 C
0921 C
0922 C
0923 C
0924 C
0925 C
0926 C
0927 C
0928 C
0929 C
0930 C
0931 C
0932 C
0933 C
0934 C
0935 C
0936 C
0937 C
0938 C
0939 C
0940 C
0941 C
0942 C
0943 C
0944 C
0945 C
0946 C
0947 C
0948 C
0949 C
0950 C
0951 C
0952 C
0953 C
0954 C
0955 C
0956 C
0957 C
0958 C
0959 C
0960 C
0961 C
0962 C
0963 C
0964 C
0965 C
0966 C
0967 C
0968 C
0969 C
0970 C
0971 C
0972 C
0973 C
0974 C
0975 C
0976 C
0977 C
0978 C
0979 C
0980 C
0981 C
0982 C
0983 C
0984 C
0985 C
0986 C
0987 C
0988 C
0989 C
0990 C
0991 C
0992 C
0993 C
0994 C
0995 C
0996 C
0997 C
0998 C
0999 C
1000 C

```

```

0001 SUBROUTINE ASK2(MAXLEN,BLKLEN)
C
C PURPOSE: TO ASK FOR THE CALCULATION BLOCK LENGTH.
C
C V.R NAME DATE COMMENTS
C #.1 J.M. KRESSE 28-JUN-79 ORIGINATION
C-----DECLARATIONS
C
0002 PARAMETER MINLEN=1,RATE=6.4
0003 INTEGER BLKLEN,MAXLEN
0004 REAL BLKTIM,MAXTIM,MINTIM
C
C-----FUNCTIONS
C
0005 TIME(INT)=FLOAT(INT)/RATE
C
C-----PROCEDURE
C
0006 MINTIM=TIME(MINLEN)
0007 MAXTIM=TIME(MAXLEN)
C
C DO UNTIL A VALID BLOCK LENGTH IS INPUT
C
0008 CONTINUE
0009 WRITE(5,20)MINLEN,MAXLEN,MINTIM,MAXTIM
0010 FORMAT(' THE CALCULATION BLOCK LENGTH MUST BE BETWEEN ',
1 15,' AND ',15,' SAMPLES.'/ ' INCLUSIVE (BETWEEN ',G11.4,
2 1 AND ',G11.4,' MS)')
3 1 HOW MANY SAMPLES PER BLOCK DO YOU WANT?')
0011 READ(5,*)BLKLEN
0012 IF((BLKLEN.LT.MINLEN).OR.(BLKLEN.GT.MAXLEN))GO TO 10
0013 CONTINUE
0014 BLKTIM=TIME(BLKLEN)
0015 WRITE(5,30)BLKTIM
0016 FORMAT(' THE BLOCK IS ',G11.4,' MS LONG.')
0017 RETURN
0018 END
30

```


TR:KS/

0001 SUBROUTINE GETSEM(S,SHAT,SHBUF,SHBUS,LUNSH,BLKLEN,SPOINT,
1 MAXBUF,BUFLEN,BPOINT,EOF,MORES)

PURPOSE: TO GET A BLOCK OF S, SHAT SAMPLES

| V.R | NAME | DATE | COMMENTS |
|-----|-------------|-----------|-------------|
| 0.1 | J.M. KRESSE | 26-JUN-79 | ORIGINATION |

DECLARATIONS

0002 PARAMETER DIGITS=5,NO=.FALSE.,YES=.TRUE.
0003 INTEGER BLKEN,BPOINT,BUFLEN,COUNT,IX,MAXBUF,S(BLKLEN)
0004 INTEGER SHUF(MAXBUF),SHAT(BLKLEN),SHBUF(MAXBUF),SPOINT
0005 LOGICAL EOF,MORES

INITIALIZATION

SPOINT=0

PROCEDURE

DO WHILE ((SPOINT.LT.BLKLEN).AND.(MORES.EQV.YES))

IF(.NOT.((SPOINT.LT.BLKLEN).AND.(MORES.EQV.YES)))GO TO 110

DO WHILE ((BPOINT.LT.BUFLEN).AND.(SPOINT.LT.BLKLEN))

IF(.NOT.((BPOINT.LT.BUFLEN).AND.(SPOINT.LT.BLKLEN)))
GO TO 30

1 BPOINT=BPOINT+1

SPOINT=SPOINT+1

S(SPOINT)=SHUF(BPOINT)

SHAT(SPOINT)=SHBUF(BPOINT)

GO TO 20

CONTINUE

CASE:((BPOINT.EQ.BUFLEN).AND.(EOF.EQV.NO))

IF(.NOT.((BPOINT.EQ.BUFLEN).AND.(EOF.EQV.NO)))GO TO 80

1 READ(LUNS,50,END=40)COUNT,
(SHUF(IX),IX=1,(COUNT/DIGITS))

1 READ(LUNSH,50,END=60)COUNT,
(SHBUF(IX),IX=1,(COUNT/DIGITS))

FORMAT(C,<BUFLEN>I<DIGITS>)

IF(COUNT.EQ.0)GO TO 60

GO TO 70

CONTINUE

EOF=YES

BUFLEN=(COUNT/DIGITS)

GO TO 70

CONTINUE

BPOINT=0

GO TO 100

CONTINUE

CASE:((BPOINT.EQ.BUFLEN).AND.(EOF.EQV.YES))

0006
0007
0008
0009
0010
0011
0012
0013
0014
0015
0016
0017
0018
0019
0020
0021
0022
0023
0024
0025
0026
0027
0028

```

FORTRAN IV-PLUS V02-51      14:37:54    15-APR-80      PAGE 2
GETSEM.FTN /TR:BLOCKS/VR

      C
0029      IF(.NOT.((BPOINT.EQ.BUFLEN).AND.(EOF.EQV.YES)))GO TO 90
0030      MORES=NO
0031      GO TO 100
0032      CONTINUE
      C
      CASE:(SPOINT.EQ.BLKLEN) (DEFAULT)
      C
      GO TO 100
      CONTINUE
      C
      GO TO 10
      CONTINUE
      RETURN
      END
0033
0034      100
0035
0036      110
0037
0038

```

215

```

FORTRAN IV-PLUS V02-51      14:38:15      15-APR-88      PAGE 2
DBC.FTN  /TR:BLOCKS/VR

0029      48      GO TO 50
0030      C      CONTINUE

0031      C      CASE:((SIGSOR.NE.B.).AND.(NOISE.NE.B.)) (DEFAULT)
0032      C      SIGSTR=DBCLIP(SIGSOR/(FLOAT(SPOINT)*REF))
0033      C      SNR=DBCLIP(SIGSOR/NOISE)
0034      50      GO TO 50
0035      C      CONTINUE
0036      C      RETURN
0037      C      END

```

TR: KS/

0001

SUBROUTINE PUTSEM(SIGSTR,SNR,LUNSCR)

C

PURPOSE: TO WRITE THE SIGNAL STRENGTH AND SNR.

C

V.R

DATE

NAME

COMMENTS

0.1

27-JUN-79

J.M. KRESSE

ORIGINATION

0.2

12-JUL-79

J.M. KRESSE

TO ADD THE SNR HEADER

C-----DECLARATIONS

C

INTEGER LUNSCR

REAL SIGSTR,SNR

0002

0003

C

C-----PROCEDURE

C

WRITE(LUNSCR,10)SIGSTR,SNR

10

FORMAT(2G12.4)

RETURN

END

0004

0005

0006

0007

```

FORTRAN IV-PLUS V02-51      14:38:45      15-APR-80      PAGE 1
CLOSER.FTN /TR:BLOCKS/WR

0001      SUBROUTINE CLOSER(LUNS,LUNSH,LUNDB,LUNSCR)
C
C      PURPOSE:  TO CLOSE THE FILES
C
C      V.R      DATE      NAME      COMMENTS
C      0.1      27-JUN-79      J.M. KRESSE      ORIGINATION
C      0.2      12-JUL-79      J.M. KRESSE      TO ADD THE SNR HEADER
C-----DECLARATIONS
C      INTEGER LUNDB,LUNS,LUNSCR,LUNSH
C-----PROCEDURE
C
C      CLOSE(UNIT=LUNS)
C      CLOSE(UNIT=LUNSH)
C      CLOSE(UNIT=LUNDB)
C      CLOSE(UNIT=LUNSCR)
C      RETURN
C      END
0002
0003
0004
0005
0006
0007
0008

```

SIG

KS/

0001 PROGRAM SNRSIG

C PURPOSE: TO PLOT LOCAL SNR VS. LOCAL SIGNAL STRENGTH

C V.R NAME DATE COMMENTS

C 0.1 J.M. KRESSE 05-JUL-79 ORIGINATION

C-----DECLARATIONS

0002 PARAMETER ENDPLT=999,LUNDAT=7,NEWORG=-3,PXMAX=8.5,PyWIND=7.5

0003 PARAMETER PYMAX=10.56,PyWIND=9.5

0004 INTEGER COUNT

0005 REAL PXVO,PyVO,RAVX,RAVXMN,RAVYMX

0006 REAL RAVY,RAVYMN,RAVYMX,SNR,SNRSEG,SUM

0007 REAL VX,VXWIND,VV,VVWIND

C-----DEFINITIONS

0008 VXWIND=PyWIND

0009 VVWIND=PXWIND

C-----INITIALIZATION

0010 CALL PLOTS(0,0,0)

0011 COUNT=0

0012 SUM=0

C-----PROCEDURE

C OPEN THE DATAFILE ON LOGICAL UNIT LUNDAT, AND READ IN THE SNR.

C CALL OPEN1(LUNDAT)

C READ(LUNDAT,*)SNR

C ASK FOR THE PLOTTING RANGES.

C CALL RANGE1(RAVXMN,RAVYMX,RAVYMN,RAVYMX)

C MOVE THE ORIGIN TO THE VIRTUAL ORIGIN.

C PXVO=PXMAX-(PXMAX-PXWIND)/2.

C PyVO=(PYMAX-PYWIND)/2.

C CALL PLOT(PXVO,PyVO,NEWORG)

C DRAW THE VIRTUAL AXES.

C CALL AXES1(RAVXMN,RAVYMX,RAVYMN,RAVYMX,VXWIND,VVWIND)

C DO UNTIL EOF

C CONTINUE

C READ(LUNDAT,*,END=20)RAVX,RAVY

C CALL RAUTOV(RAVX,RAVYMN,RAVYMX,VXWIND,VX)

C CALL RAUTOV(RAVY,RAVYMN,RAVYMX,VVWIND,VV)

C COUNT=COUNT+1

C SUM=SUM+VY

0020 10

0021

0022

0023

0024

0025

```

FORTRAN IV-PLUS V82-S1      14:38:58      15-APR-88      PAGE 2
SNRSIG.FTN /TR:BLOCKS/VR

      CALL POINT1(VX,VY)
      GO TO 18
      CONTINUE
      COMPUTE SNRSEG.
      SUM=SUM/FLOAT(COUNT)
      SNRSEG=SUM*(RAVVMX-RAVVMN)/VVWIND+RAVVMN
      LABEL THE PLOT.
      CALL LABEL1(RAVVMN,RAVVMX,VXWIND,VVWIND,SNR,SNRSEG)
      END THE PLOTTING AND CLOSE THE DATAFILE.
      CALL PLOT(B.,B.,ENDPLT)
      CLOSE(UNIT=LUNDAT)
      STOP
      END

```



```

0001 SUBROUTINE OPEN1(LUNDAT)
C
C PURPOSE: TO OPEN THE DATA FILE ON LOGICAL UNIT LUNDAT.
C
C V.R NAME DATE COMMENTS
C 8.1 J.M. KRESSE 85-JUL-79 ORIGINATION
C
C-----DECLARATIONS
C
0002 PARAMETER DIMENS=33,NUL=8
0003 BYTE DATAFI(DIMENS)
0004 INTEGER LUNDAT
C
C-----PROCEDURE
C
C ASK FOR DATAFILENAME.
C
0005 WRITE(5,18)
0006 FORMAT(' WHAT IS THE NAME OF THE DATAFILE TO BE PLOTTED?')
0007 READ(5,28)DATAFI
0008 FORMAT(<DIMENS>A1)
C
C INSERT END-OF-STRING MARK.
C
0009 DATAFI(DIMENS)=NUL
C
C OPEN THE DATAFILE ON LOGICAL UNIT LUNDAT.
C
0010 OPEN(UNIT=LUNDAT,NAME=DATAFI,TYPE='OLD',SHARED,READONLY)
0011 RETURN
0012 END

```

```

FORTRAN IV-PLUS V02-51      14:39:25      15-APR-80      PAGE 1
RANGE1.FTN
0001      SUBROUTINE RANGE1(RAVXNM,RAVXM,RAVYMN,RAVYM)
0002      PURPOSE: TO ASK FOR THE PLOTTING RANGES.
0003      V.R      NAME      DATE      COMMENTS
0004      0.1      J.M. KRESSE      09-JUL-79      ORIGINATION
0005      C-----DECLARATIONS
0006      C
0007      C
0008      C
0009      C
0010      C
0011      C
0012      C
0013      C
0014      C
0015      C
0016      C
0017      C
0018      C
0019      C
0020      C
0021      C
0022      C
0023      C
0024      C
0025      C
0026      C
0027      C
0028      C
0029      C
0030      C
0031      C
0032      C
0033      C
0034      C
0035      C
0036      C
0037      C
0038      C
0039      C
0040      C
0041      C
0042      C
0043      C
0044      C
0045      C
0046      C
0047      C
0048      C
0049      C
0050      C
0051      C
0052      C
0053      C
0054      C
0055      C
0056      C
0057      C
0058      C
0059      C
0060      C
0061      C
0062      C
0063      C
0064      C
0065      C
0066      C
0067      C
0068      C
0069      C
0070      C
0071      C
0072      C
0073      C
0074      C
0075      C
0076      C
0077      C
0078      C
0079      C
0080      C
0081      C
0082      C
0083      C
0084      C
0085      C
0086      C
0087      C
0088      C
0089      C
0090      C
0091      C
0092      C
0093      C
0094      C
0095      C
0096      C
0097      C
0098      C
0099      C
0100      C
0101      C
0102      C
0103      C
0104      C
0105      C
0106      C
0107      C
0108      C
0109      C
0110      C
0111      C
0112      C
0113      C
0114      C
0115      C
0116      C
0117      C
0118      C
0119      C
0120      C
0121      C
0122      C
0123      C
0124      C
0125      C
0126      C
0127      C
0128      C
0129      C
0130      C
0131      C
0132      C
0133      C
0134      C
0135      C
0136      C
0137      C
0138      C
0139      C
0140      C
0141      C
0142      C
0143      C
0144      C
0145      C
0146      C
0147      C
0148      C
0149      C
0150      C
0151      C
0152      C
0153      C
0154      C
0155      C
0156      C
0157      C
0158      C
0159      C
0160      C
0161      C
0162      C
0163      C
0164      C
0165      C
0166      C
0167      C
0168      C
0169      C
0170      C
0171      C
0172      C
0173      C
0174      C
0175      C
0176      C
0177      C
0178      C
0179      C
0180      C
0181      C
0182      C
0183      C
0184      C
0185      C
0186      C
0187      C
0188      C
0189      C
0190      C
0191      C
0192      C
0193      C
0194      C
0195      C
0196      C
0197      C
0198      C
0199      C
0200      C
0201      C
0202      C
0203      C
0204      C
0205      C
0206      C
0207      C
0208      C
0209      C
0210      C
0211      C
0212      C
0213      C
0214      C
0215      C
0216      C
0217      C
0218      C
0219      C
0220      C
0221      C
0222      C
0223      C
0224      C
0225      C
0226      C
0227      C
0228      C
0229      C
0230      C
0231      C
0232      C
0233      C
0234      C
0235      C
0236      C
0237      C
0238      C
0239      C
0240      C
0241      C
0242      C
0243      C
0244      C
0245      C
0246      C
0247      C
0248      C
0249      C
0250      C
0251      C
0252      C
0253      C
0254      C
0255      C
0256      C
0257      C
0258      C
0259      C
0260      C
0261      C
0262      C
0263      C
0264      C
0265      C
0266      C
0267      C
0268      C
0269      C
0270      C
0271      C
0272      C
0273      C
0274      C
0275      C
0276      C
0277      C
0278      C
0279      C
0280      C
0281      C
0282      C
0283      C
0284      C
0285      C
0286      C
0287      C
0288      C
0289      C
0290      C
0291      C
0292      C
0293      C
0294      C
0295      C
0296      C
0297      C
0298      C
0299      C
0300      C
0301      C
0302      C
0303      C
0304      C
0305      C
0306      C
0307      C
0308      C
0309      C
0310      C
0311      C
0312      C
0313      C
0314      C
0315      C
0316      C
0317      C
0318      C
0319      C
0320      C
0321      C
0322      C
0323      C
0324      C
0325      C
0326      C
0327      C
0328      C
0329      C
0330      C
0331      C
0332      C
0333      C
0334      C
0335      C
0336      C
0337      C
0338      C
0339      C
0340      C
0341      C
0342      C
0343      C
0344      C
0345      C
0346      C
0347      C
0348      C
0349      C
0350      C
0351      C
0352      C
0353      C
0354      C
0355      C
0356      C
0357      C
0358      C
0359      C
0360      C
0361      C
0362      C
0363      C
0364      C
0365      C
0366      C
0367      C
0368      C
0369      C
0370      C
0371      C
0372      C
0373      C
0374      C
0375      C
0376      C
0377      C
0378      C
0379      C
0380      C
0381      C
0382      C
0383      C
0384      C
0385      C
0386      C
0387      C
0388      C
0389      C
0390      C
0391      C
0392      C
0393      C
0394      C
0395      C
0396      C
0397      C
0398      C
0399      C
0400      C
0401      C
0402      C
0403      C
0404      C
0405      C
0406      C
0407      C
0408      C
0409      C
0410      C
0411      C
0412      C
0413      C
0414      C
0415      C
0416      C
0417      C
0418      C
0419      C
0420      C
0421      C
0422      C
0423      C
0424      C
0425      C
0426      C
0427      C
0428      C
0429      C
0430      C
0431      C
0432      C
0433      C
0434      C
0435      C
0436      C
0437      C
0438      C
0439      C
0440      C
0441      C
0442      C
0443      C
0444      C
0445      C
0446      C
0447      C
0448      C
0449      C
0450      C
0451      C
0452      C
0453      C
0454      C
0455      C
0456      C
0457      C
0458      C
0459      C
0460      C
0461      C
0462      C
0463      C
0464      C
0465      C
0466      C
0467      C
0468      C
0469      C
0470      C
0471      C
0472      C
0473      C
0474      C
0475      C
0476      C
0477      C
0478      C
0479      C
0480      C
0481      C
0482      C
0483      C
0484      C
0485      C
0486      C
0487      C
0488      C
0489      C
0490      C
0491      C
0492      C
0493      C
0494      C
0495      C
0496      C
0497      C
0498      C
0499      C
0500      C
0501      C
0502      C
0503      C
0504      C
0505      C
0506      C
0507      C
0508      C
0509      C
0510      C
0511      C
0512      C
0513      C
0514      C
0515      C
0516      C
0517      C
0518      C
0519      C
0520      C
0521      C
0522      C
0523      C
0524      C
0525      C
0526      C
0527      C
0528      C
0529      C
0530      C
0531      C
0532      C
0533      C
0534      C
0535      C
0536      C
0537      C
0538      C
0539      C
0540      C
0541      C
0542      C
0543      C
0544      C
0545      C
0546      C
0547      C
0548      C
0549      C
0550      C
0551      C
0552      C
0553      C
0554      C
0555      C
0556      C
0557      C
0558      C
0559      C
0560      C
0561      C
0562      C
0563      C
0564      C
0565      C
0566      C
0567      C
0568      C
0569      C
0570      C
0571      C
0572      C
0573      C
0574      C
0575      C
0576      C
0577      C
0578      C
0579      C
0580      C
0581      C
0582      C
0583      C
0584      C
0585      C
0586      C
0587      C
0588      C
0589      C
0590      C
0591      C
0592      C
0593      C
0594      C
0595      C
0596      C
0597      C
0598      C
0599      C
0600      C
0601      C
0602      C
0603      C
0604      C
0605      C
0606      C
0607      C
0608      C
0609      C
0610      C
0611      C
0612      C
0613      C
0614      C
0615      C
0616      C
0617      C
0618      C
0619      C
0620      C
0621      C
0622      C
0623      C
0624      C
0625      C
0626      C
0627      C
0628      C
0629      C
0630      C
0631      C
0632      C
0633      C
0634      C
0635      C
0636      C
0637      C
0638      C
0639      C
0640      C
0641      C
0642      C
0643      C
0644      C
0645      C
0646      C
0647      C
0648      C
0649      C
0650      C
0651      C
0652      C
0653      C
0654      C
0655      C
0656      C
0657      C
0658      C
0659      C
0660      C
0661      C
0662      C
0663      C
0664      C
0665      C
0666      C
0667      C
0668      C
0669      C
0670      C
0671      C
0672      C
0673      C
0674      C
0675      C
0676      C
0677      C
0678      C
0679      C
0680      C
0681      C
0682      C
0683      C
0684      C
0685      C
0686      C
0687      C
0688      C
0689      C
0690      C
0691      C
0692      C
0693      C
0694      C
0695      C
0696      C
0697      C
0698      C
0699      C
0700      C
0701      C
0702      C
0703      C
0704      C
0705      C
0706      C
0707      C
0708      C
0709      C
0710      C
0711      C
0712      C
0713      C
0714      C
0715      C
0716      C
0717      C
0718      C
0719      C
0720      C
0721      C
0722      C
0723      C
0724      C
0725      C
0726      C
0727      C
0728      C
0729      C
0730      C
0731      C
0732      C
0733      C
0734      C
0735      C
0736      C
0737      C
0738      C
0739      C
0740      C
0741      C
0742      C
0743      C
0744      C
0745      C
0746      C
0747      C
0748      C
0749      C
0750      C
0751      C
0752      C
0753      C
0754      C
0755      C
0756      C
0757      C
0758      C
0759      C
0760      C
0761      C
0762      C
0763      C
0764      C
0765      C
0766      C
0767      C
0768      C
0769      C
0770      C
0771      C
0772      C
0773      C
0774      C
0775      C
0776      C
0777      C
0778      C
0779      C
0780      C
0781      C
0782      C
0783      C
0784      C
0785      C
0786      C
0787      C
0788      C
0789      C
0790      C
0791      C
0792      C
0793      C
0794      C
0795      C
0796      C
0797      C
0798      C
0799      C
0800      C
0801      C
0802      C
0803      C
0804      C
0805      C
0806      C
0807      C
0808      C
0809      C
0810      C
0811      C
0812      C
0813      C
0814      C
0815      C
0816      C
0817      C
0818      C
0819      C
0820      C
0821      C
0822      C
0823      C
0824      C
0825      C
0826      C
0827      C
0828      C
0829      C
0830      C
0831      C
0832      C
0833      C
0834      C
0835      C
0836      C
0837      C
0838      C
0839      C
0840      C
0841      C
0842      C
0843      C
0844      C
0845      C
0846      C
0847      C
0848      C
0849      C
0850      C
0851      C
0852      C
0853      C
0854      C
0855      C
0856      C
0857      C
0858      C
0859      C
0860      C
0861      C
0862      C
0863      C
0864      C
0865      C
0866      C
0867      C
0868      C
0869      C
0870      C
0871      C
0872      C
0873      C
0874      C
0875      C
0876      C
0877      C
0878      C
0879      C
0880      C
0881      C
0882      C
0883      C
0884      C
0885      C
0886      C
0887      C
0888      C
0889      C
0890      C
0891      C
0892      C
0893      C
0894      C
0895      C
0896      C
0897      C
0898      C
0899      C
0900      C
0901      C
0902      C
0903      C
0904      C
0905      C
0906      C
0907      C
0908      C
0909      C
0910      C
0911      C
0912      C
0913      C
0914      C
0915      C
0916      C
0917      C
0918      C
0919      C
0920      C
0921      C
0922      C
0923      C
0924      C
0925      C
0926      C
0927      C
0928      C
0929      C
0930      C
0931      C
0932      C
0933      C
0934      C
0935      C
0936      C
0937      C
0938      C
0939      C
0940      C
0941      C
0942      C
0943      C
0944      C
0945      C
0946      C
0947      C
0948      C
0949      C
0950      C
0951      C
0952      C
0953      C
0954      C
0955      C
0956      C
0957      C
0958      C
0959      C
0960      C
0961      C
0962      C
0963      C
0964      C
0965      C
0966      C
0967      C
0968      C
0969      C
0970      C
0971      C
0972      C
0973      C
0974      C
0975      C
0976      C
0977      C
0978      C
0979      C
0980      C
0981      C
0982      C
0983      C
0984      C
0985      C
0986      C
0987      C
0988      C
0989      C
0990      C
0991      C
0992      C
0993      C
0994      C
0995      C
0996      C
0997      C
0998      C
0999      C
1000      C

```

```

PLUM-2-5-14: 18 3-AF 3
AXES1.FTN /TR:BLOCKS/VR
SUBROUTINE AXES1(RAWXMN,RAWVMX,RAVVMN,RAWVMX,VXWIND,VYWIND)
C
C PURPOSE: TO DRAW THE AXES.
C
C V.R NAME DATE COMMENTS
C 8.1 J.M. KRESSE 85-JUL-79 ORIGINATION
C-----DECLARATIONS
C
C PARAMETER CCW=1,CV=-1,LMASK1=-21846,LMASK2=-38584
C PARAMETER MINDIV=5,XUNDIV=18.,YUNDIV=5.
C INTEGER DIR,NX,NY,VNX,VNY
C REAL RAWXMN,RAWVMX,RAVVMN,RAWVMX,VANGLE,VAXLEN,VDV
C REAL VFVAL,VX,VXD,VXFACT,VXWIND,VY,VYD,VYFACT,VYWIND,XD,YD
C
C-----PROCEDURE
C
C DRAW THE X AXIS.
C
C VX=8.
C CALL RAWTOV(8.,RAWXMN,RAWVMX,VYWIND,VY)
C DIR=CV
C VAXLEN=VYWIND
C VANGLE=8.
C VFVAL=RAWXMN
C VDV=XUNDIV
C VXFACT=VYWIND/((RAWXMN-RAWXMN)/XUNDIV)
C CALL VAXIS(VX,VY,DIR,VAXLEN,VANGLE,VFVAL,VDV,VXFACT)
C
C DRAW THE Y AXIS.
C
C CALL RAWTOV(8.,RAWXMN,RAWVMX,VXWIND,VX)
C VY=8.
C DIR=CCW
C VAXLEN=VYWIND
C VANGLE=98.
C VFVAL=RAVVMN
C VDV=YUNDIV
C VYFACT=VYWIND/((RAVVMX-RAVVMN)/YUNDIV)
C CALL VAXIS(VX,VY,DIR,VAXLEN,VANGLE,VFVAL,VDV,VYFACT)
C
C DRAW THE MAJOR GRID.
C
C VX=8.
C VY=8.
C VNX=IFIX((RAWXMN-RAWXMN)/XUNDIV)
C VXD=VXFACT
C VNY=IFIX((RAVVMX-RAVVMN)/YUNDIV)
C VYD=VYFACT
C CALL ROTATE(VX,VY,VANGLE,X,Y,ANGLE)
C CALL ROTATE(VXD,VYD,VANGLE,XD,YD,ANGLE)
C NX=VNX
C NY=VNY
C CALL GRID(X,Y,NX,XD,Y,YD,LMASK1)
C
C DRAW THE MINOR GRID.

```

FORTRAM IV-PLUS V02-51
AXES1.FTM /TR:BLOCKS/VR

14:39:40 15-APR-80

PAGE 2

C

```
0036 NX=NX*MINDIV
0037 XD=XD/FLOAT(MINDIV)
0038 NV=NV*MINDIV
0039 YD=YD/FLOAT(MINDIV)
0040 CALL GRID(X,Y,NX,XD,NV,YD,LMASK2)
0041 RETURN
0042 END
```

```

0001 SUBROUTINE LABEL1(RAWXMN,RAWXMX,VXWIND,VYWIND,VVWIND,SNR,SNRSEG)
C
C PURPOSE: TO LABEL THE PLOT.
C
C V.R DATE NAME COMMENTS
C 0.1 10-JUL-79 J.M. KRESSE ORIGINATION
C
C PARAMETER HAFLIN=40,HEIGHT=.175,MARGIN=.5
C PARAMETER NCHAR=36,NCHR3D=26,VANGLE=0.
C INTEGER ASCI(3),ASC2(3)
C INTEGER IX,LINE1((NCHAR+1)/2),LINE2(HAFLIN),LINE3((NCHR3D+1)/2)
C INTEGER MXCHAR,NCHAR1,NCHAR2,NCHAR3,STR1(2),STR2(5)
C REAL ANGLE,RAWXMN,RAWXMX,SNR,SNRSEG,VX,VXWIND,VY,VYWIND,X,Y
C EQUIVALENCE (LINE3(1),STR1(1)),(LINE3(3),ASCI(1))
C EQUIVALENCE (LINE3(6),STR2(1)),(LINE3(11),ASC2(1))
C
C-----DEFINITION
C
C DATA LINE1 /'LO','CA','L','SN','R','(D','B)','V','S',
C 1 'L','OC','AL','S','IG','NA','L','(D','B)'/
C DATA STR1 /'SN','R='/'
C DATA STR2 /'S','NR','SE','G='/'
C
C-----FUNCTION
C
C START(NCHAR)=YAXIS*(VXWIND-YAXIS)/.-((FLOAT(NCHAR)*HEIGHT)/2.
C
C-----PROCEDURE
C
C LOCATE THE V-AXIS.
C
C CALL RAWTOV(0.,RAWXMN,RAWXMX,VXWIND,VYAXIS)
C
C CALCULATE THE MAXIMUM NUMBER OF CHARACTERS PERMITTED.
C
C MXCHAR=MIN(2*HAFLIN,IFIX((VXWIND-YAXIS-2.*MARGIN)/HEIGHT))
C NCHAR1=MIN(MXCHAR,NCHAR)
C NCHAR3=MIN(MXCHAR,NCHR3D)
C
C ASK FOR THE LABELING INFORMATION.
C
C CONTINUE
C WRITE(5,20)MXCHAR
C FORMAT(' HOW DO YOU WANT TO LABEL THE PLOT? /
C 1 ('.13,' CHARACTERS, MAXIMUM)')
C READ(5,30)NCHAR2,(LINE2(IX),IX=1,(NCHAR2+1)/2)
C 30 FORMAT(Q,(NCHAR2+1)/2>A2)
C IF(NCHAR2.GT.MXCHAR)GO TO 10
C CONTINUE
C
C START THE FIRST LINE AT THE TOP, CENTERED BETWEEN THE
C Y-AXIS AND THE RIGHT EDGE.
C
C VX=START(NCHAR1)
C VY=VYWIND-MARGIN-HEIGHT
C CALL ROTATE(VX,VY,VANGLE,X,Y,ANGLE)

```

```

FORTBAN IV-PLUS V02-51      14:39:55      15-APR-80      PAGE 2
LABEL1.FTN /TR:BLOCKS/WR

0020      C      CALL SYMBOL(X,Y,HEIGHT,LINE1,ANGLE,NCHAR1)
          C      PLOT THE SECOND LINE.
          C
0029      C      VX=START(NCHAR2)
0030      C      VV=VV-2."HEIGHT
0031      C      CALL ROTATE(VX,VV,VANGLE,X,Y,ANGLE)
0032      C      CALL SYMBOL(X,Y,HEIGHT,LINE2,ANGLE,NCHAR2)
          C
          C      GENERATE THE THIRD LINE.
          C
0033      C      ENCODE(6,40,ASC1)SNR
0034      C      ENCODE(6,40,ASC2)SNRSEG
0035      C      FORMAT(F6.2)
          C
          C      PLOT THE THIRD LINE.
          C
0036      C      VX=START(NCHAR3)
0037      C      VV=VV-2."HEIGHT
0038      C      CALL ROTATE(VX,VV,VANGLE,X,Y,ANGLE)
0039      C      CALL SYMBOL(X,Y,HEIGHT,LINE3,ANGLE,NCHAR3)
0040      C      RETURN
0041      C      END

```

```

TRAN-PU 15-1 38
VAXIS.FTN /TR:BLOCKS/VR
0001 SUBROUTINE VAXIS(VX,VV,DIR,VAXLEN,VANGLE,VFVAL,VDV,FACT)
C
C PURPOSE: TO DRAW A VIRTUAL AXIS.
C
C V.R NAME DATE COMMENTS
C 0.1 J.M. KRESSE 5-JUL-79 ORIGINATION
C-----DECLARATIONS
C
C INTEGER DIR
0002 REAL ANGLE,AXLEN,FACT,VANGLE,VAXLEN,VDV,VFVAL,VX,VV,X,Y
0003 C-----PROCEDURE
C
C ROTATE
C
C CALL ROTATE(VX,VV,VANGLE,X,Y,ANGLE)
C
C FIX ABSOLUTE
C
C X=X/FACT
C Y=Y/FACT
C AXLEN=VAXLEN/FACT
C CALL FACTOR(FACT)
C CALL AXIS(X,Y,DIR,AXLEN,ANGLE,VFVAL,VDV)
C CALL FACTOR(1.)
C RETURN
C END
0005
0006
0007
0008
0009
0010
0011
0012

```

```

FORTRAN IV-PLUS V#2-51      16:54:17    15-APR-88      PAGE 1
POINT1.FTN  /TR:BLOCKS/VR

0001      SUBROUTINE POINT1(VX,VY)
C
C      PURPOSE: TO PLOT A POINT CORRESPONDING TO (RAWX,RAWY).
C
C      V.R      NAME      DATE      COMMENTS
C      0.1      J.M. KRESSE    10-JUL-79    ORIGINATION
C-----DECLARATIONS
C      PARAMETER HEIGHT=.075,ITEXT=11,NC=-1,VANGLE=0.
C      REAL ANGLE,VX,VY,X,Y
C-----PROCEDURE
C      CALL ROTATE(VX,VY,VANGLE,X,V,ANGLE)
C      CALL SYMBOL(X,Y,HEIGHT,ITEXT,ANGLE,NC)
C      RETURN
C      END
0002
0003
0004
0005
0006
0007

```


0001

SUBROUTINE RAWTOV(RAW,RAVMN,RAVMX,VWIND,V)

PURPOSE: TO CONVERT THE RAW DATA TO VIRTUAL DATA.

| V.R | DATE | NAME | COMMENTS |
|-----|-----------|-------------|-------------|
| 0.1 | 10-JUL-79 | J.M. KRESSE | ORIGINATION |

DECLARATIONS

REAL RAW,RAVMN,RAVMX,V,VWIND

PROCEDURE

V=AMIN(1.,AMAX(0.,(RAW-RAVMN)/(RAVMX-RAVMN)))*VWIND
 RETURN
 END

0002

0003
 0004
 0005

```

FORTRAN IV-PLUS V02-51      14:48:53      15-APR-88      PAGE 1
ROTATE.FTN /TR:BLOCKS/VR

0001      SUBROUTINE ROTATE(VX,VV,VANGLE,X,Y,ANGLE)
C
C      PURPOSE:  TO ROTATE THE VIRTUAL DATA TO PHYSICAL DATA.
C
C      V.R      DATE      NAME      COMMENTS
C      0.1      18-JUL-79      J.M. KRESSE      ORIGINATION
C-----DECLARATIONS
C      REAL ANGLE,VANGLE,VX,VV,X,Y
C-----PROCEDURE
C
C      X=-VV
C      Y=VX
C      ANGLE=VANGLE+98.
C      RETURN
C      END
0002
0003
0004
0005
0006
0007

```

```

0001      PROGRAM BFCALC
0002      C
0003      C PURPOSE: TO CALCULATE THE BLOCK SIGNAL STRENGTH AND THE BLOCK
0004      C AVERAGE BUFFER LENGTH.
0005      C
0006      C V.R      DATE      NAME      COMMENTS
0007      C 0.1      22-JUL-79      J.M. KRESSE      ORIGINATION
0008      C-----DECLARATIONS
0009      C
0010      C PARAMETER LUNIN=1,LUNOUT=2,LUNTI=5,MAXSIG=60.,MINSIG=-60.
0011      C BYTE FNAME(NAMLEN)
0012      C INTEGER BLKLEN,ICNT
0013      C LOGICAL EOF
0014      C REAL BUFAVG,BUFENT,BUFSUM,SIG,SIGSTR,SIGSUM
0015      C
0016      C-----PROCEDURE
0017      C
0018      C ASK FOR THE INPUT FILENAME, AND OPEN THE INPUTFILE ON LOGICAL
0019      C UNIT LUNIN.
0020      C
0021      C WRITE(LUNTI,10)
0022      C FORMAT(' WHAT IS THE INPUT FILENAME?')
0023      C READ(LUNTI,20)FNAME
0024      C FORMAT(<NAMLEN>A1)
0025      C FNAME(NAMLEN)=NUL
0026      C OPEN(UNIT=LUNIN,NAME=FNAME,TYPE='OLD',READONLY,SHARED)
0027      C
0028      C ASK FOR THE OUTPUT FILENAME, AND OPEN THE OUTPUTFILE ON LOGICAL
0029      C UNIT LUNOUT.
0030      C
0031      C WRITE(LUNTI,30)
0032      C FORMAT(' WHAT IS THE OUTPUT FILENAME?')
0033      C READ(LUNTI,20)FNAME
0034      C FNAME(NAMLEN)=NUL
0035      C OPEN(UNIT=LUNOUT,NAME=FNAME,TYPE='NEW',CARRIAGECONTROL='LIST')
0036      C
0037      C ASK FOR THE AVERAGING BLOCKLENGTH.
0038      C
0039      C WRITE(LUNTI,40)
0040      C FORMAT(' WHAT BLOCKLENGTH DO YOU WANT TO USE?')
0041      C READ(LUNTI,*)BLKLEN
0042      C EOF=NO
0043      C
0044      C DO UNTIL (EOF.EQV.YES)
0045      C
0046      C CONTINUE
0047      C SIGSUM=0.
0048      C BUFSUM=0.
0049      C ICNT=0
0050      C
0051      C DO UNTIL ((ICNT.EQ.BLKLEN).OR.(EOF.EQV.YES))
0052      C
0053      C CONTINUE
0054      C READ(LUNIN,*,END=70)SIG,BUFENT
0055      C
0056      C
0057      C
0058      C
0059      C
0060      C
0061      C
0062      C
0063      C
0064      C
0065      C
0066      C
0067      C
0068      C
0069      C
0070      C
0071      C
0072      C
0073      C
0074      C
0075      C
0076      C
0077      C
0078      C
0079      C
0080      C
0081      C
0082      C
0083      C
0084      C
0085      C
0086      C
0087      C
0088      C
0089      C
0090      C
0091      C
0092      C
0093      C
0094      C
0095      C
0096      C
0097      C
0098      C
0099      C
0100      C
0101      C
0102      C
0103      C
0104      C
0105      C
0106      C
0107      C
0108      C
0109      C
0110      C
0111      C
0112      C
0113      C
0114      C
0115      C
0116      C
0117      C
0118      C
0119      C
0120      C
0121      C
0122      C
0123      C
0124      C
0125      C
0126      C
0127      C
0128      C
0129      C
0130      C
0131      C
0132      C
0133      C
0134      C
0135      C
0136      C
0137      C
0138      C
0139      C
0140      C
0141      C
0142      C
0143      C
0144      C
0145      C
0146      C
0147      C
0148      C
0149      C
0150      C
0151      C
0152      C
0153      C
0154      C
0155      C
0156      C
0157      C
0158      C
0159      C
0160      C
0161      C
0162      C
0163      C
0164      C
0165      C
0166      C
0167      C
0168      C
0169      C
0170      C
0171      C
0172      C
0173      C
0174      C
0175      C
0176      C
0177      C
0178      C
0179      C
0180      C
0181      C
0182      C
0183      C
0184      C
0185      C
0186      C
0187      C
0188      C
0189      C
0190      C
0191      C
0192      C
0193      C
0194      C
0195      C
0196      C
0197      C
0198      C
0199      C
0200      C
0201      C
0202      C
0203      C
0204      C
0205      C
0206      C
0207      C
0208      C
0209      C
0210      C
0211      C
0212      C
0213      C
0214      C
0215      C
0216      C
0217      C
0218      C
0219      C
0220      C
0221      C
0222      C
0223      C
0224      C
0225      C
0226      C
0227      C
0228      C
0229      C
0230      C
0231      C
0232      C
0233      C
0234      C
0235      C
0236      C
0237      C
0238      C
0239      C
0240      C
0241      C
0242      C
0243      C
0244      C
0245      C
0246      C
0247      C
0248      C
0249      C
0250      C
0251      C
0252      C
0253      C
0254      C
0255      C
0256      C
0257      C
0258      C
0259      C
0260      C
0261      C
0262      C
0263      C
0264      C
0265      C
0266      C
0267      C
0268      C
0269      C
0270      C
0271      C
0272      C
0273      C
0274      C
0275      C
0276      C
0277      C
0278      C
0279      C
0280      C
0281      C
0282      C
0283      C
0284      C
0285      C
0286      C
0287      C
0288      C
0289      C
0290      C
0291      C
0292      C
0293      C
0294      C
0295      C
0296      C
0297      C
0298      C
0299      C
0300      C
0301      C
0302      C
0303      C
0304      C
0305      C
0306      C
0307      C
0308      C
0309      C
0310      C
0311      C
0312      C
0313      C
0314      C
0315      C
0316      C
0317      C
0318      C
0319      C
0320      C
0321      C
0322      C
0323      C
0324      C
0325      C
0326      C
0327      C
0328      C
0329      C
0330      C
0331      C
0332      C
0333      C
0334      C
0335      C
0336      C
0337      C
0338      C
0339      C
0340      C
0341      C
0342      C
0343      C
0344      C
0345      C
0346      C
0347      C
0348      C
0349      C
0350      C
0351      C
0352      C
0353      C
0354      C
0355      C
0356      C
0357      C
0358      C
0359      C
0360      C
0361      C
0362      C
0363      C
0364      C
0365      C
0366      C
0367      C
0368      C
0369      C
0370      C
0371      C
0372      C
0373      C
0374      C
0375      C
0376      C
0377      C
0378      C
0379      C
0380      C
0381      C
0382      C
0383      C
0384      C
0385      C
0386      C
0387      C
0388      C
0389      C
0390      C
0391      C
0392      C
0393      C
0394      C
0395      C
0396      C
0397      C
0398      C
0399      C
0400      C
0401      C
0402      C
0403      C
0404      C
0405      C
0406      C
0407      C
0408      C
0409      C
0410      C
0411      C
0412      C
0413      C
0414      C
0415      C
0416      C
0417      C
0418      C
0419      C
0420      C
0421      C
0422      C
0423      C
0424      C
0425      C
0426      C
0427      C
0428      C
0429      C
0430      C
0431      C
0432      C
0433      C
0434      C
0435      C
0436      C
0437      C
0438      C
0439      C
0440      C
0441      C
0442      C
0443      C
0444      C
0445      C
0446      C
0447      C
0448      C
0449      C
0450      C
0451      C
0452      C
0453      C
0454      C
0455      C
0456      C
0457      C
0458      C
0459      C
0460      C
0461      C
0462      C
0463      C
0464      C
0465      C
0466      C
0467      C
0468      C
0469      C
0470      C
0471      C
0472      C
0473      C
0474      C
0475      C
0476      C
0477      C
0478      C
0479      C
0480      C
0481      C
0482      C
0483      C
0484      C
0485      C
0486      C
0487      C
0488      C
0489      C
0490      C
0491      C
0492      C
0493      C
0494      C
0495      C
0496      C
0497      C
0498      C
0499      C
0500      C
0501      C
0502      C
0503      C
0504      C
0505      C
0506      C
0507      C
0508      C
0509      C
0510      C
0511      C
0512      C
0513      C
0514      C
0515      C
0516      C
0517      C
0518      C
0519      C
0520      C
0521      C
0522      C
0523      C
0524      C
0525      C
0526      C
0527      C
05
```

```

FORTRAN IV-PLUS V02-51      14:41:02      15-APR-88      PAGE 2
BFCALC.FTN      /TR:BLOCKS/WR

0029      SIGSUM=SIGSUM+ABS(SIG)
0030      BUFSUM=BUFSUM+8UFCNT
0031      ICNT=ICNT+1
0032      GO TO 80
0033      CONTINUE
0034      EOF=YES
0035      GO TO 80
0036      CONTINUE
0037      IF(.NOT.((ICNT.EQ.BLKLEN).OR.(EOF.EQV.YES)))GO TO 60
0038      CONTINUE
0039      IF(ICNT.EQ.0)GO TO 120
0040      IF(SIGSUM.EQ.0.)GO TO 90
0041      SIGSTR=AMIN1(MAXSIG,AMAX1(MINSIG,(20.*
1      ALOG10(SIGSUM/(FLOAT(ICNT)*REF))))
0042      GO TO 100
0043      CONTINUE
0044      SIGSTR=MINSIG
0045      GO TO 100
0046      CONTINUE
0047      BUFAVG=BUFSUM/FLOAT(ICNT)
0048      WRITE(LUNOUT,110)SIGSTR,BUFAVG
0049      110    FORMAT(F7.2,F10.2)
0050      GO TO 120
0051      CONTINUE
0052      IF(EOF.EQV.NO)GO TO 50
0053      CONTINUE
0054      CLOSE(UNIT=LUNIN)
0055      CLOSE(UNIT=LUNOUT)
0056      STOP
0057      END

```

COMMAND FILE TO D B JT.7

BFLOT/CP/FP.BFLOT/NOSP=BFLOT,OPEN1,RANGE2,LABEL2,AXES2,VAXIS.
POINT1,RAVTOV,ROTATE,(1.54)PEPLIB.OLB/LB

UNITS=7
ACTFIL=7
//

```

FORTRAN IV-PLUS V82-51      14:41:38  15-APR-88      PAGE 1
BFLOT.FTN  /TRIBLOCKS/WR

0001  PROGRAM BFLOT
C
C  PURPOSE:  TO PLOT THE AVERAGE BUFFER LENGTH VS. LOCAL SIGNAL
C  STRENGTH.
C
C  V.R      DATE      NAME      COMMENTS
C  8.1      23-JUL-79  J.M. KRESSE  ORIGINATION
C
C-----DECLARATIONS
C
0002  PARAMETER ENDPLT=999,LUNDAT=7,NEWORG=-3,PXMAX=8.5,PXWIND=7.5
0003  PARAMETER PYMAX=10.56,PYWIND=9.5
0004  REAL PXVO,PYVO,RAWX,RAWXMN,RAWXMX
0005  REAL RAWY,RAWYMN,RAWYMX
0006  REAL VX,VXWIND,VV,VVWIND
C
C-----DEFINITIONS
C
0007  VXWIND=PYWIND
0008  VVWIND=PXWIND
C
C-----INITIALIZATION
C
0009  CALL PLOTS(8,8,8)
C
C-----PROCEDURE
C
C  OPEN THE DATAFILE ON LOGICAL UNIT LUNDAT.
C
0010  CALL OPEN1(LUNDAT)
C
C  ASK FOR THE PLOTTING RANGES.
C
0011  CALL RANGE2(RAWXMN,RAWXMX,RAWYMN,RAWYMX)
C
C  MOVE THE ORIGIN TO THE VIRTUAL ORIGIN.
C
0012  PXVO=PXMAX-(PXMAX-PXWIND)/2.
0013  PYVO=(PYMAX-PYWIND)/2.
0014  CALL PLOT(PXVO,PYVO,NEWORG)
C
C  DRAW THE VIRTUAL AXES.
C
0015  CALL AXES2(RAWXMN,RAWXMX,RAWYMN,RAWYMX,VXWIND,VVWIND)
C
C  DO UNTIL EOF
C
0016  CONTINUE
0017  READ(LUNDAT,*,END=28)RAWX,RAWY
0018  CALL RAWTOV(RAWX,RAWXMN,RAWXMX,RAWYMN,RAWYMX,VXWIND,VX)
0019  CALL RAWTOV(RAWY,RAWYMN,RAWYMX,RAWYMN,RAWYMX,VVWIND,VV)
0020  CALL POINT1(VX,VV)
0021  GO TO 18
0022  CONTINUE
C
C  LABEL THE PLOT.

```

FORTAN IV-PLUS VB2-51 14:41:38 15-APR-88
 PLOTTING / TR-CKS

PAGE 2

```

C
BB23      CALL LABEL2(RAWXMN,RAWXMX,VXWIND,VVWIND)
C
C          END THE PLOTTING AND CLOSE THE DATAFILE.
C
BB24      CALL PLOT(S,B,.ENDPLT)
BB25      CLOSE(UNIT=LUNDAT)
BB26      STOP
BB27      END
  
```

```

FORTRAN IV-PLUS V02-51      14:41:43      15-APR-80      PAGE 1
RANGE2.FTN      /TR:BLOCKS/VR

0001      C
0002      C
0003      C
0004      C
0005      C
0006      C
0007      C
0008      C
0009      C
0010      C
0011      C
0012      C
0013      C
0014      C
0015      C
0016      C
0017      C

SUBROUTINE RANGE2(RAVXMN,RAVYMX,RAVYMN,RAVYMX)
PURPOSE: TO ASK FOR THE PLOTTING RANGES.
V.R      DATE      NAME      COMMENTS
0.1      23-JUL-79      J.M. KRESSE      ORIGINATION
C-----DECLARATIONS
PARAMETER DEFYMN=-10.,DEFYMX=60.,DEFYMN=-2000.,DEFYMX=500.
INTEGER NCHARS
REAL RAVXMN,RAVYMX,RAVYMN,RAVYMX
C-----PROCEDURE
ASK FOR THE DESIRED RANGE.
WRITE(5,10)DEFYMN,DEFYMX,DEFYMN,DEFYMX
FORMAT(' PLEASE ENTER THE RANGES IN THE X AND Y DIRECTIONS.
1 RESPECTIVELY. '/
2 '(FOR THE DEFAULT VALUES OF '.F6.2.', '.F5.2.', '.F8.2.
3 '.F6.2.', HIT RETURN.))
READ(5,20)NCHARS,RAVXMN,RAVYMX,RAVYMN,RAVYMX
FORMAT(0,4G14.7)
IF(NCHARS.NE.0)GO TO 30
RAVXMN=DEFYMN
RAVYMX=DEFYMX
RAVYMN=DEFYMN
RAVYMX=DEFYMX
GO TO 30
CONTINUE
RETURN
END

```


0001 SUBROUTINE AXES2(RAWXMN,RAWXMX,RAWYMN,RAWYMX,VXWIND,VYWIND)

0002 PURPOSE: TO DRAW THE AXES.

0003 V.R DATE NAME COMMENTS
0004 0.1 23-JUL-79 J.M. KRESSE ORIGINATION

0005 C-----DECLARATIONS

0006 PARAMETER CCW=1,CV=-1,LMASK1=-21846,LMASK2=-30584
PARAMETER MINDIV=5,XUNDIV=10.,YUNDIV=500.
INTEGER DIR,NX,NY,VNX,VNY
REAL RAWXMN,RAWXMX,RAWYMN,RAWYMX,VANGLE,VAXLEN,VDV
REAL VFVAL,VX,VXD,VXFACT,VXWIND,VY,VYD,VYFACT,VYWIND,XD,YD

0007 C-----PROCEDURE

0008 DRAW THE X AXIS.

0009 VX=0.
0010 CALL RAWTOV(0.,RAWYMN,RAWYMX,VYWIND,VY)
0011 DIR=CV
0012 VAXLEN=VXWIND
0013 VANGLE=0.
0014 VFVAL=RAWYMN
0015 VDV=YUNDIV
0016 VXFACT=VXWIND/((RAWYMX-RAWYMN)/XUNDIV)
0017 CALL VAXIS(VX,VY,DIR,VAXLEN,VANGLE,VFVAL,VDV,VXFACT)

0018 DRAW THE Y AXIS.

0019 CALL RAWTOV(0.,RAWXMN,RAWXMX,VXWIND,VX)
0020 VY=0.
0021 DIR=CCW
0022 VAXLEN=VYWIND
0023 VANGLE=90.
0024 VFVAL=RAWXMN
0025 VDV=XUNDIV
0026 VYFACT=VYWIND/((RAWYMX-RAWYMN)/YUNDIV)
0027 CALL VAXIS(VX,VY,DIR,VAXLEN,VANGLE,VFVAL,VDV,VYFACT)

0028 DRAW THE MAJOR GRID.

0029 VX=0.
0030 VY=0.
0031 VNX=FIX((RAWYMX-RAWYMN)/XUNDIV)
0032 VXD=VXFACT
0033 VNY=FIX((RAWYMX-RAWYMN)/YUNDIV)
0034 VVD=VYFACT
0035 CALL ROTATE(VX,VY,VANGLE,X.Y,ANGLE)
0036 CALL ROTATE(VXD,VYD,VANGLE,XD,YD,ANGLE)
0037 NX=VNX
0038 NY=VNY
0039 CALL GRID(X,Y,NX,XD,NY,YD,LMASK1)

0040 DRAW THE MINOR GRID.

PAGE 2

15-APR-88

14:41:55

FORTRAN IV-PLUS V82-51
AXES2.FTN /TR:BLOCKS/WR

C

```
0036 NX=NX*MINDIV
0037 XD=XD/FLOAT(MINDIV)
0038 NY=NY*MINDIV
0039 YD=YD/FLOAT(MINDIV)
0040 CALL GRID(X,Y,NX,XD,NY,YD,LMASK2)
0041 RETURN
0042 END
```

 SUBROUTINE LABEL2(RAVXMN,RAVXMX,VXWIND,VYVWIND)

PURPOSE: TO LABEL THE PLOT.

V.R DATE NAME COMMENTS
 8.1 23-JUL-79 J.M. KRESSE ORIGINATION

PARAMETER HAFLIN=48,HEIGHT=.175,MARGIN=.5
 PARAMETER NCHAR=36,VANGLE=8.
 INTEGER IX,LINE1((NCHAR+1)/2),LINE2(HAFLIN)
 INTEGER MXCHAR,NCHAR1,NCHAR2
 REAL ANGLE,RAVXMN,RAVXMX,VX,VXWIND,VY,VYVWIND,X,Y

C-----DEFINITION

DATA LINE1 /'AV','G','BU','F','LE','NG','TH','V','S',
 1 'L','OC','AL','S','IG','NA','L','D','B'//

C-----FUNCTION

START(NCHAR)=YAXIS+(VXWIND-YAXIS)/2.-(FLOAT(NCHAR)*HEIGHT)/2.

C-----PROCEDURE

LOCATE THE Y-AXIS.

CALL RANTOV(8.,RAVXMN,RAVXMX,VXWIND,YAXIS)

CALCULATE THE MAXIMUM NUMBER OF CHARACTERS PERMITTED.

MXCHAR=MIN(2*HAFLIN,ifix((VXWIND-YAXIS-2.*MARGIN)/HEIGHT))
 NCHAR1=MIN(MXCHAR,NCHAR)

ASK FOR THE LABELING INFORMATION.

CONTINUE

WRITE(5,28)MXCHAR
 FORMAT(' HOW DO YOU WANT TO LABEL THE PLOT?'/
 1 ('13.' CHARACTERS. MAXIMUM)'
 READ(5,38)NCHAR2,(LINE2(IX),IX=1,(NCHAR2+1)/2)
 FORMAT(0,(<(NCHAR2+1)/2>A2)
 IF(NCHAR2.GT.MXCHAR)GO TO 18
 CONTINUE

START THE FIRST LINE AT THE TOP, CENTERED BETWEEN THE
 Y-AXIS AND THE RIGHT EDGE.

VX=START(NCHAR1)
 VY=VYVWIND-MARGIN-HEIGHT
 CALL ROTATE(VX,VY,VANGLE,X,Y,ANGLE)
 CALL SYMBOL(X,Y,HEIGHT,LINE1,ANGLE,NCHAR1)
 PLOT THE SECOND LINE.

VX=START(NCHAR2)
 VY=VY-2.*HEIGHT

PAGE 2

15-APR-88

14:42:14

PORTMAN IV-PLUS V02-51
LABEL2.FTN /TR:BLOCKS/VR

```
0025 CALL ROTATE(VX,VY,VANGLE,X,Y,ANGLE)
0026 CALL SYMBOL(X,Y,HEIGHT,LINE2,ANGLE,NCHAR2)
0027 RETURN
0028 END
```



```
1      COMMAND FILE TO BUILD BDFPL.TSK
1
1      BDFPL/CP/FP.BDFPL/NOSP-BDFPL.OPEN1.RANGE3.LABEL3.AXES3.VAXIS.
1      POINT1.RAOTOV.ROTATE.(1.54)PEPL18.OLB/LB
1      /
1      UNITS=7
1      ACTFIL=7
1      //
```

```

0001      PROGRAM 8FDFPL
C
C      PURPOSE:  TO PLOT THE DIFFERENCE IN THE AVERAGE BUFFER LENGTH
C                VS. THE LOCAL SIGNAL STRENGTH.
C
C      V.R      DATE      NAME      COMMENTS
C      0.1      23-JUL-79      J.M. KRESSE      ORIGINATION
C
C-----DECLARATIONS
C
0002      PARAMETER ENDPLT=999,LUNDAT=7,NEWORG=-3,PXMAX=8.5,PXWIND=7.5
0003      PARAMETER PYMAX=10.56,PYWIND=9.5
0004      REAL PXVO,PYVO,RAWX,RAWXMN,RAWXMX
0005      REAL RAVY,RAVYMN,RAVYMX
0006      REAL VX,VXWIND,VY,VYWIND
C
C-----DEFINITIONS
C
0007      VXWIND=PYWIND
0008      VYWIND=PXWIND
C
C-----INITIALIZATION
C
0009      CALL PLOTS(0,0,0)
C
C-----PROCEDURE
C
C      OPEN THE DATAFILE ON LOGICAL UNIT LUNDAT.
C
0010      CALL OPEN1(LUNDAT)
C
C      ASK FOR THE PLOTTING RANGES.
C
0011      CALL RANGE3(RAWXMN,RAWXMX,RAVYMN,RAVYMX)
C
C      MOVE THE ORIGIN TO THE VIRTUAL ORIGIN.
C
0012      PXVO=PXMAX-(PXMAX-PXWIND)/2.
0013      PYVO=(PYMAX-PYWIND)/2.
0014      CALL PLOT(PXVO,PYVO,NEWORG)
C
C      DRAW THE VIRTUAL AXES.
C
0015      CALL AXES3(RAWXMN,RAWXMX,RAVYMN,RAVYMX,VXWIND,VYWIND)
C
C      DO UNTIL EOF
C
0016      CONTINUE
0017      READ(LUNDAT,*,END=20)RAWX,RAVY
0018      CALL RAVTOV(RAWX,RAVYMN,RAWXMX,VXWIND,VX)
0019      CALL RAVTOV(RAVY,RAVYMN,RAVYMX,VYWIND,VY)
0020      CALL POINT1(VX,VY)
C
C      GO TO 10
0021      CONTINUE
0022      LABEL THE PLOT.
C

```

FORTTRAN IV-PLUS V02-01 14:42:54 15-APR-80 PAGE 2
BFDPL.FTN /TR:BLOCKS/WR
C
0023 CALL LABEL3(RAVXMN,RAVYMN,VXWIND,VYWIND)
C
C END THE PLOTTING AND CLOSE THE DATAFILE.
C
0024 CALL PLOT(B..B..ENDPLT)
0025 CLOSE(UNIT=LUNDAT)
0026 STOP
0027 END


```

0001 SUBROUTINE RANGE3(RAVXMN,RAVXXMX,RAVYMN,RAVYMX)
C
C PURPOSE: TO ASK FOR THE PLOTTING RANGES.
C
C V.R DATE NAME COMMENTS
C 0.1 23-JUL-79 J.M. KRESSE ORIGINATION
C-----DECLARATIONS
C
0002 PARAMETER DEFYMN=-10.,DEFXXM=60.,DEFYMN=-130.,DEFYMX=120.
0003 INTEGER NCHARS
0004 REAL RAVXMN,RAVXXMX,RAVYMN,RAVYMX
C-----PROCEDURE
C
C ASK FOR THE DESIRED RANGE.
C
0005 WRITE(5,10)DEFYMN,DEFXXM,DEFYMN,DEFYMX
0006 FORMAT(' PLEASE ENTER THE RANGES IN THE X AND Y DIRECTIONS,
1 RESPECTIVELY.'/
2 '(FOR THE DEFAULT VALUES OF ',F6.2,', ',F5.2,', ',F8.2,
3 ', ',F6.2,', HIT RETURN.))
0007 READ(5,20)NCHARS,RAVXMN,RAVXXMX,RAVYMN,RAVYMX
0008 FORMAT(0,4G14.7)
0009 IF(NCHARS.NE.0)GO TO 30
0010 RAVXMN=DEFYMN
0011 RAVXXM=DEFXXM
0012 RAVYMN=DEFYMN
0013 RAVYMX=DEFYMX
0014 GO TO 30
0015 CONTINUE
0016 RETURN
0017 END

```

```

FORTRAN IV-PLUS V02-51      14:43:20      15-APR-80      PAGE 1
AXES3.FTN      /TR:BLOCKS/WR

0001  SUBROUTINE AXES3(RAWXMN,RAWXMX,RAWYMN,RAWYMX,VXWIND,VYWIND)
      C
      C      PURPOSE: TO DRAW THE AXES.
      C
      C      V.R      DATE      NAME      COMMENTS
      C      0.1      23-JUL-79      J.M. KRESSE      ORIGINATION
      C
      C-----DECLARATIONS
      C
      C      PARAMETER CCW=1,CW=-1,LMASK1=-21846,LMASK2=-30584
      C      PARAMETER MINDIV=5,XUNDIV=10.,YUNDIV=50.
      C      INTEGER DIR,NX,NY,VNX,VNY
      C      REAL RAWXMN,RAWXMX,RAWYMN,RAWYMX,VANGLE,VAXLEN,VDV
      C      REAL VFVAL,VX,VXD,VXFACT,VXWIND,VV,VYD,VYFACT,VYWIND,XD,YD
      C
      C-----PROCEDURE
      C
      C      DRAW THE X AXIS.
      C
      C
      C      VX=0.
      C      CALL RAUTOV(0.,RAWXMN,RAWXMX,VYWIND,VV)
      C      DIR=CCW
      C      VAXLEN=VXWIND
      C      VANGLE=0.
      C      VFVAL=RAWXMN
      C      VDV=XUNDIV
      C      VXFACT=VXWIND/((RAWXMX-RAWXMN)/XUNDIV)
      C      CALL VAXIS(VX,VV,DIR,VAXLEN,VANGLE,VFVAL,VDV,VXFACT)
      C
      C      DRAW THE Y AXIS.
      C
      C
      C      CALL RAUTOV(0.,RAWXMN,RAWXMX,VXWIND,VX)
      C      VV=0.
      C      DIR=CCW
      C      VAXLEN=VYWIND
      C      VANGLE=90.
      C      VFVAL=RAWYMN
      C      VDV=YUNDIV
      C      VYFACT=VYWIND/((RAWYMX-RAWYMN)/YUNDIV)
      C      CALL VAXIS(VX,VV,DIR,VAXLEN,VANGLE,VFVAL,VDV,VYFACT)
      C
      C      DRAW THE MAJOR GRID.
      C
      C
      C      VX=0.
      C      VV=0.
      C      VNX=FIX((RAWXMX-RAWXMN)/XUNDIV)
      C      VXD=VXFACT
      C      VNY=FIX((RAWYMX-RAWYMN)/YUNDIV)
      C      VYD=VYFACT
      C      CALL ROTATE(VX,VV,VANGLE,X,Y,ANGLE)
      C      CALL ROTATE(VXD,VYD,VANGLE,XD,YD,ANGLE)
      C      NX=VNX
      C      NY=VNY
      C      CALL GRID(X,Y,NX,XD,NY,YD,LMASK1)
      C
      C      DRAW THE MINOR GRID.
      C

```

FORTRAN IV-PLUS V02-51 14:43:20 15-APR-80

PAGE 2

MESS /TR

C

```

0036 NX=NX*MINDIV
0037 XD=XD/FLOAT(MINDIV)
0038 NV=NV*MINDIV
0039 YD=YD/FLOAT(MINDIV)
0040 CALL GRID(X,Y,NX,XD,NV,YD,LMASK2)
0041 RETURN
0042 END

```

```

FORTAN IV-PLUS V02-51      14:43:38      15-APR-80      PAGE 1
LABEL3.FTN /TR:BLOCKS/WR

0001 SUBROUTINE LABEL3(RAWXMN,RAWXMX,VXWIND,VYWIND)
C
C PURPOSE: TO LABEL THE PLOT.
C
C V.R DATE NAME COMMENTS
C 0.1 .3-JUL-79 J.M. KRESSE ORIGINATION
C
C PARAMETER HAFIN=40,HEIGHT=.175,MARGIN=.5
C PARAMETER NCHAR=36,VANGLE=0.
C INTEGER IX,LINE1((NCHAR+1)/2),LINE2(HAFIN)
C INTEGER MXCHAR,NCHAR1,NCHAR2
C REAL ANGLE,RAWXMN,RAWXMX,VX,VXWIND,VY,VYWIND,X,Y
C
C-----DEFINITION
C
0002 DATA LINE1 /'DI','F','AV','G','BU','FF','ER','V','S','
0003 1 'L','OC','AL','S','IG','NA','L','(D','B)'/
C
C-----FUNCTION
C
0004 START(NCHAR)=YAXIS+(VXWIND-YAXIS)/2.-((FLOAT(NCHAR)*HEIGHT)/2.
C
C-----PROCEDURE
C
C LOCATE THE Y-AXIS.
C
0005 CALL RAVTOV(0.,RAWXMN,RAWXMX,VXWIND,YAXIS)
C
C CALCULATE THE MAXIMUM NUMBER OF CHARACTERS PERMITTED.
C
0006 MXCHAR=MIN(2*HAFIN,IFX((VXWIND-YAXIS-2.*MARGIN)/HEIGHT))
0007 NCHAR1=MIN(MXCHAR,NCHAR)
C
C ASK FOR THE LABELING INFORMATION.
C
0008 CONTINUE
0009 WRITE(5,20)MXCHAR
0010 FORMAT(' HOW DO YOU WANT TO LABEL THE PLOT?'/
0011 1 ' (.13,' CHARACTERS, MAXIMUM)')
0012 READ(5,30)NCHAR2,(LINE2(IX),IX=1,(NCHAR2+1)/2)
0013 FORMAT(0,(NCHAR2+1)/2>A2)
0014 IF(NCHAR2.GT.MXCHAR)GO TO 10
0015 CONTINUE
C
C START THE FIRST LINE AT THE TOP, CENTERED BETWEEN THE
C Y-AXIS AND THE RIGHT EDGE.
C
0016 VX=START(NCHAR1)
0017 VY=VYWIND-MARGIN-HEIGHT
0018 CALL ROTATE(VX,VY,VANGLE,X,Y,ANGLE)
0019 CALL SYMBOL(X,V,HEIGHT,LINE1,ANGLE,NCHAR1)
C
C PLOT THE SECOND LINE.
C
0020 VX=START(NCHAR2)
0021 VY=VY-2.*HEIGHT
0022
0023
0024

```

TR:BLOCKS/WR
 LABEL3.FTN
 1:38 15-80
 CALL ROTATE(VX,VY,VANGLE,X,Y,ANGLE)
 CALL SYMBOL(X,Y,HEIGHT,LINE2,ANGLE,NCHAR2)
 RETURN
 END

0025
 0026
 0027
 0028

APPENDIX C

PLOTTING PROGRAM

This program is used to plot a data file or the difference of two data file on a Versaplot 07 system. It employs the Versaplot-07 PPEP Software Package.

Two options are provided by this program:

1. It can plot a whole data file. The ranges of x-axis and y-axis are specified by the user through a terminal.
2. It can plot a number of sections of a data file. The starting location and number of sections can be specified by the user through a terminal. However, the size of a section is fixed to 1600 samples.

Data must be stored in 1615 format with a standard format header card. Details are shown in the program listing.

```
*****
*
*   VERSATEC DATAFILE PLOTTING PROGRAM
*
*****
```

```
MAVLIN VEH
DATE:6/28/79
PROGRAM NAME:  DATAPLOT.FTM
```

THIS PROGRAM IS USED TO PLOT A SPEECH SAMPLE FILE ON
 ON A VERSATEC PRINTER/PLOTTER. THE FORMAT FORMAT FOR
 THE SPEECH SAMPLES IS 1615.

THERE ARE TWO FUNCTIONS IN THIS PROGRAM:

1. PLOT A DATA FILE
2. PLOT THE DIFFERENCE BETWEEN TWO DATA FILES

ALSO, TWO OPTIONS ARE AVAILABLE AS FOLLOWS:

1. PLOT THE WHOLE DATA FILE.
2. PLOT PART OF DATA FILE.

```
DIMENSION X(1682),Y(1682)
LOGICAL=1 NAME1(32),NAME2(32),SYM1(88),SYM2(88)
INTEGER=2 H1(48),H2(48),X1(16),X2(16),X3(16)
```

ASK FOR THE CHOICE

```
TYPE *, ' WHICH FUNCTION DO YOU WANT?'
TYPE *, ' 1 - PLOT A DATA FILE'
TYPE *, ' 2 - PLOT THE DIFFERENCE BETWEEN TWO FILE (FILE1-FILE2)'
ACCEPT *, IOPT
TYPE *, ' DO YOU WANT TO PLOT THE WHOLE DATA FILE?(1=YES,2=NO)'
```

```
ACCEPT *, IOPT2
IF(IOPT2.EQ. 1)GO TO 78
TYPE *, ' WHAT IS THE STARTING SAMPLE NUMBER?'
ACCEPT *, ISTART
TYPE *, ' NUMBER OF WINDOWS YOU WANT TO PLOT?'
TYPE *, ' ( WINDOW SIZE = 1688 SAMPLES )'
ACCEPT *, NWINDOW
IF(IOPT.EQ. 1) GO TO 28
IF(IOPT.NE. 2) GO TO 18
```

FUNCTION 2

```
TYPE *, ' INPUT 1ST FILE NAME?'
ACCEPT 1888,NAME1
NAME1(32)=8
OPEN(UNIT=9,NAME=NAME1,READONLY,SHARED,TYPE='OLD')
READ(9,1881)NSENT1,IRATE1,NSAMP1,IUPER1,ILOWR1,NTERM1,H1
TYPE *, ' 1ST FILE HEADER IS'
TYPE 1881,NSENT1,IRATE1,NSAMP1,IUPER1,ILOWR1,NTERM1,H1
TYPE *, ' THE LABEL OF THIS FILE FOR THE PLOT?'
TYPE *, ' ( 88 CHARACTERS, MAXIMUM )'
ACCEPT 1886,SYM1
```

FORTRAN IV-PLUS V82-51 11:17:25 31-MAR-88 PAGE 2
DATAPLOT.FTN /TR:BLOCKS/VR

```

8828 TYPE ' ', INPUT 2ND FILE NAME?
8829 ACCEPT 1888,NAME2
8830 NAME2(32)=8
8831 OPEN(UNIT=7,NAME=NAME2,READONLY,SHARED,TYPE='OLD')
8832 READ(7,1881)NSENT2,IRATE2,NSAMP2,IUPER2,ILOWR2,INTERM2,H2
8833 TYPE ' ', 2ND FILE HEADER IS.
8834 TYPE 1881,NSENT2,IRATE2,NSAMP2,IUPER2,ILOWR2,INTERM2,H2
8835 TYPE ' ', THE LABEL OF THIS FILE FOR THE PLOT?
8836 TYPE ' ', ( 88 CHARACTERS, MAXIMUM )
8837 ACCEPT 1886,SYM2

```

C
C
C

```

8838 PRODUCE A SCRATCH FILE
8839 OPEN(UNIT=8,NAME='SCRATCH.DAT',TYPE='SCRATCH')
8840 READ(9,1884,END=48)X1
8841 READ(7,1884,END=48)X2

```

58
38

```

8842 DO 38 1=1,16
8843 X3(1)=X1(1)-X2(1)
8844 WRITE(8,1884)X3
8845 GO TO 58

```

48

```

8846 REVIND 8
8847 GO TO 68
8848 TYPE ' ', INPUT FILE NAME?
8849 ACCEPT 1888,NAME1

```

28

```

8850 NAME1(32)=8
8851 OPEN(UNIT=8,NAME=NAME1,READONLY,SHARED,TYPE='OLD')
8852 TYPE ' ', INPUT FILE HEADER IS.
8853 READ(8,1881)NSENT,IRATE,NSAMP,IUPER,ILOWR,INTERMS,H1
8854 TYPE 1881,NSENT,IRATE,NSAMP,IUPER,ILOWR,INTERMS,H1
8855 TYPE ' ', THE LABEL OF THIS FILE FOR THE PLOT?
8856 TYPE ' ', ( 88 CHARACTERS, MAXIMUM )
8857 ACCEPT 1886,SYM1

```

```

8858 FORMAT(1615,15X,48A1)
8859 FORMAT(32A1)
8860 FORMAT(1615)
8861 FORMAT(88A1)

```

1881

1888
1884
1886

C
C
C

PLOTTING ROUTINE

```

8861 TYPE ' ', THE WIDTH FOR X-AXIS IS "X-WIDTH" INCHES FOR 1688 SAMPLES'
8862 TYPE ' ', THE WIDTH FOR Y-AXIS IS 8 INCHES.
8863 TYPE ' ', INPUT WINDOW SIZE?(XMAX,XMIN,YMAX,YMIN)'
8864 TYPE ' ', E.G. 144. 8. 18. 8.
8865 ACCEPT ' ',XMAX,XMIN,YMAX,YMIN
8866 TYPE ' ', X-WIDTH,X-SCALING FACTOR,Y-STARTING VALUE,Y-SCALING FACTOR ?'
8867 TYPE ' ', E.G. 16. 188. -2488. 688. ( FOR ONE FILE )
8868 TYPE ' ', E.G. 16. 188. -688. 158. ( FOR THE DIFFERENCE FILE )
8869 ACCEPT ' ',XWIDTH,XSCAL,YSTART,YSCAL
8870 IF(IOPT2 .EQ. 1)GO TO 88
8871 ISAMP=16
8872 IF(1SAMP .GE. 1SAMP)GO TO 188
8873 ISAMP=1SAMP+16
8874 READ(8,1884)X1
8875 GO TO 98

```

98

```

8876 SAM=1SAMP
8877 SAMP=1SAMP

```



```

8878      FA=SAMP
8879      DO 188 I=1,1688,16
8880      READ(8,1884)X1
8881      DO 288 J=1,16
8882      V(I+J-1)=X1(J)
8883      CONTINUE
8884      DO 388 I=1,1688
8885      FA=FA+1.
8886      X(I)=FA
8887      CALL PLOTS(8.,8.,8.)
8888      CALL PLOT(8.,8.,-3)
8889      CALL WINDOW(XMAX,XMIN,YMAX,YMIN)
8890      CALL SYMBOL(2.,9.,8.14,SYM1,8.,88)
8891      IF(IOPT.EQ. 1) GO TO 655
8892      CALL SYMBOL(2.,8.5,8.14,SYM2,8.,88)
8893      CALL PLOT(1.,1.,-3)
8894      X(1681)=SAMP
8895      X(1682)=XSCAL
8896      Y(1681)=YSTART
8897      Y(1682)=YSCAL
8898      CALL AXIS(8.,8.,13HSAMPLE NUMBER,-13,XWIDTH,8.,
1      X(1681),X(1682))
1      CALL AXIS(8.,8.,9SHAMPLITUDE,9,8.,98.,
      Y(1681),Y(1682))
8900      CALL LINE(X,Y,1688,1,8,8)
8901      IF(IOPT2.EQ. 1) GO TO 581
8902      NVINDO=NVINDO-1
8903      IF(NVINDO.EQ. 8) GO TO 118
8904      XTEMP=X(1688)
8905      YTEMP=Y(1688)
8906      ICOUNT=8.
8907      READ(8,1884,END=788)X1
8908      DO 333 J=1,16
8909      ICOUNT=ICOUNT+1
8910      V(1688)=X1(J)
8911      IF(ICOUNT.NE. 1688) GO TO 688
8912      GO TO 988
8913      DO 988 I=ICOUNT+1,1688
8914      IEND=1
8915      V(I)=8.
8916      DO 444 I=1,1688
8917      FA=FA+1.
8918      X(I)=FA
8919      XX=(X(1)-X(1681))/X(1682)
8920      VV=(V(1)-V(1681))/V(1682)
8921      XTEMP1=(XTEMP-X(1681))/X(1682)
8922      YTEMP1=(YTEMP-Y(1681))/Y(1682)
8923      CALL PLOT(XTEMP1,YTEMP1,3)
8924      CALL PLOT(XX,VV,2)
8925      X(1681)=X(1681)+1688.
8926      CALL PLOT(16.,8.,-3)
8927      CALL AXIS(8.,8.,13HSAMPLE NUMBER,-13,16.,8.,
1      X(1681),X(1682))
1      IF(IEND.NE. 1) GO TO 588
8928      CALL LINE(X,Y,1688,1,8,8)
8929      CALL PLOT(8.,8.,999)
8930
8931
8932
8933
8934
8935
8936
8937
8938
8939
8940
8941
8942
8943
8944
8945
8946
8947
8948
8949
8950
8951
8952
8953
8954
8955
8956
8957
8958
8959
8960
8961
8962
8963
8964
8965
8966
8967
8968
8969
8970
8971
8972
8973
8974
8975
8976
8977
8978
8979
8980
8981
8982
8983
8984
8985
8986
8987
8988
8989
8990
8991
8992
8993
8994
8995
8996
8997
8998
8999
9000
9001
9002
9003
9004
9005
9006
9007
9008
9009
9010
9011
9012
9013
9014
9015
9016
9017
9018
9019
9020
9021
9022
9023
9024
9025
9026
9027
9028
9029
9030
9031
9032
9033
9034
9035
9036
9037
9038
9039
9040
9041
9042
9043
9044
9045
9046
9047
9048
9049
9050
9051
9052
9053
9054
9055
9056
9057
9058
9059
9060
9061
9062
9063
9064
9065
9066
9067
9068
9069
9070
9071
9072
9073
9074
9075
9076
9077
9078
9079
9080
9081
9082
9083
9084
9085
9086
9087
9088
9089
9090
9091
9092
9093
9094
9095
9096
9097
9098
9099
9100
9101
9102
9103
9104
9105
9106
9107
9108
9109
9110
9111
9112
9113
9114
9115
9116
9117
9118
9119
9120
9121
9122
9123
9124
9125
9126
9127
9128
9129
9130
9131
9132
9133
9134
9135
9136
9137
9138
9139
9140
9141
9142
9143
9144
9145
9146
9147
9148
9149
9150
9151
9152
9153
9154
9155
9156
9157
9158
9159
9160
9161
9162
9163
9164
9165
9166
9167
9168
9169
9170
9171
9172
9173
9174
9175
9176
9177
9178
9179
9180
9181
9182
9183
9184
9185
9186
9187
9188
9189
9190
9191
9192
9193
9194
9195
9196
9197
9198
9199
9200
9201
9202
9203
9204
9205
9206
9207
9208
9209
9210
9211
9212
9213
9214
9215
9216
9217
9218
9219
9220
9221
9222
9223
9224
9225
9226
9227
9228
9229
9230
9231
9232
9233
9234
9235
9236
9237
9238
9239
9240
9241
9242
9243
9244
9245
9246
9247
9248
9249
9250
9251
9252
9253
9254
9255
9256
9257
9258
9259
9260
9261
9262
9263
9264
9265
9266
9267
9268
9269
9270
9271
9272
9273
9274
9275
9276
9277
9278
9279
9280
9281
9282
9283
9284
9285
9286
9287
9288
9289
9290
9291
9292
9293
9294
9295
9296
9297
9298
9299
9300
9301
9302
9303
9304
9305
9306
9307
9308
9309
9310
9311
9312
9313
9314
9315
9316
9317
9318
9319
9320
9321
9322
9323
9324
9325
9326
9327
9328
9329
9330
9331
9332
9333
9334
9335
9336
9337
9338
9339
9340
9341
9342
9343
9344
9345
9346
9347
9348
9349
9350
9351
9352
9353
9354
9355
9356
9357
9358
9359
9360
9361
9362
9363
9364
9365
9366
9367
9368
9369
9370
9371
9372
9373
9374
9375
9376
9377
9378
9379
9380
9381
9382
9383
9384
9385
9386
9387
9388
9389
9390
9391
9392
9393
9394
9395
9396
9397
9398
9399
9400
9401
9402
9403
9404
9405
9406
9407
9408
9409
9410
9411
9412
9413
9414
9415
9416
9417
9418
9419
9420
9421
9422
9423
9424
9425
9426
9427
9428
9429
9430
9431
9432
9433
9434
9435
9436
9437
9438
9439
9440
9441
9442
9443
9444
9445
9446
9447
9448
9449
9450
9451
9452
9453
9454
9455
9456
9457
9458
9459
9460
9461
9462
9463
9464
9465
9466
9467
9468
9469
9470
9471
9472
9473
9474
9475
9476
9477
9478
9479
9480
9481
9482
9483
9484
9485
9486
9487
9488
9489
9490
9491
9492
9493
9494
9495
9496
9497
9498
9499
9500
9501
9502
9503
9504
9505
9506
9507
9508
9509
9510
9511
9512
9513
9514
9515
9516
9517
9518
9519
9520
9521
9522
9523
9524
9525
9526
9527
9528
9529
9530
9531
9532
9533
9534
9535
9536
9537
9538
9539
9540
9541
9542
9543
9544
9545
9546
9547
9548
9549
9550
9551
9552
9553
9554
9555
9556
9557
9558
9559
9560
9561
9562
9563
9564
9565
9566
9567
9568
9569
9570
9571
9572
9573
9574
9575
9576
9577
9578
9579
9580
9581
9582
9583
9584
9585
9586
9587
9588
9589
9590
9591
9592
9593
9594
9595
9596
9597
9598
9599
9600
9601
9602
9603
9604
9605
9606
9607
9608
9609
9610
9611
9612
9613
9614
9615
9616
9617
9618
9619
9620
9621
9622
9623
9624
9625
9626
9627
9628
9629
9630
9631
9632
9633
9634
9635
9636
9637
9638
9639
9640
9641
9642
9643
9644
9645
9646
9647
9648
9649
9650
9651
9652
9653
9654
9655
9656
9657
9658
9659
9660
9661
9662
9663
9664
9665
9666
9667
9668
9669
9670
9671
9672
9673
9674
9675
9676
9677
9678
9679
9680
9681
9682
9683
9684
9685
9686
9687
9688
9689
9690
9691
9692
9693
9694
9695
9696
9697
9698
9699
9700
9701
9702
9703
9704
9705
9706
9707
9708
9709
9710
9711
9712
9713
9714
9715
9716
9717
9718
9719
9720
9721
9722
9723
9724
9725
9726
9727
9728
9729
9730
9731
9732
9733
9734
9735
9736
9737
9738
9739
9740
9741
9742
9743
9744
9745
9746
9747
9748
9749
9750
9751
9752
9753
9754
9755
9756
9757
9758
9759
9760
9761
9762
9763
9764
9765
9766
9767
9768
9769
9770
9771
9772
9773
9774
9775
9776
9777
9778
9779
9780
9781
9782
9783
9784
9785
9786
9787
9788
9789
9790
9791
9792
9793
9794
9795
9796
9797
9798
9799
9800
9801
9802
9803
9804
9805
9806
9807
9808
9809
9810
9811
9812
9813
9814
9815
9816
9817
9818
9819
9820
9821
9822
9823
9824
9825
9826
9827
9828
9829
9830
9831
9832
9833
9834
9835
9836
9837
9838
9839
9840
9841
9842
9843
9844
9845
9846
9847
9848
9849
9850
9851
9852
9853
9854
9855
9856
9857
9858
9859
9860
9861
9862
9863
9864
9865
9866
9867
9868
9869
9870
9871
9872
9873
9874
9875
9876
9877
9878
9879
9880
9881
9882
9883
9884
9885
9886
9887
9888
9889
9890
9891
9892
9893
9894
9895
9896
9897
9898
9899
9900
9901
9902
9903
9904
9905
9906
9907
9908
9909
9910
9911
9912
9913
9914
9915
9916
9917
9918
9919
9920
9921
9922
9923
9924
9925
9926
9927
9928
9929
9930
9931
9932
9933
9934
9935
9936
9937
9938
9939
9940
9941
9942
9943
9944
9945
9946
9947
9948
9949
9950
9951
9952
9953
9954
9955
9956
9957
9958
9959
9960
9961
9962
9963
9964
9965
9966
9967
9968
9969
9970
9971
9972
9973
9974
9975
9976
9977
9978
9979
9980
9981
9982
9983
9984
9985
9986
9987
9988
9989
9990
9991
9992
9993
9994
9995
9996
9997
9998
9999

```

PAGE 4

31-MAR-85

11:17:25

FORTRAN IV-PLUS V02-51
DATAPLOT.FTN /TR:BLOCKS/VR9131 CALL CLOSE(8)
9132 STOP
9133 END

APPENDIX D

PDP-11 D/A PROGRAMS

This set of D/A programs is used for a digital-to-analog converting operation employing the DATEL's ST-PDP device which is an analog I/O module for DEC PDP-11 minicomputers. The ST-PDP device has three different modes:

1. Program Control Interface
2. Interrupt Serviced Interface
3. Direct Memory Address

The set of D/A programs uses the Program Control Interface mode.

The set consists of six modules:

1. DA.CMD ——— The indirect command file to execute the D/A converting operation.
2. DASP.FTN ——— The Fortran file to output a speech data file to the D/A converter.
3. DARAMP.FTN ——— The Fortran file to output a ramp function to the D/A converter in order to check the timing and operation.
4. DA64.MAC ——— The MACRO-11 assembly language subroutine for 6400 samples per second sampling frequency.
5. DA80.MAC ——— The MACRO-11 assembly language subroutine for 8000 samples per second sampling frequency.
6. COMMON.MAC ——— The MACRO-11 assembly language file used to build a common device block inside the PDP-11 operating system.

This set of programs allows a user to output a data file from disk to the D/A converter. Data must be stored in 1615 format with a standard format header card. Details are shown in the program listing.

A common device block has to be built the first time the D/A device is used. The way to build is shown below. Further details can be found in the I/O Driver Reference Manual of the PDP-11 RMS-11M operating system.

1. Logon a privileged UIC: For future reference, use UIC = [3,1]

2. > MAC COMMON = COMMON

(underline means the prompt of the computer)

3. > SET /UIC = [1,1]

4. > TKB

TKB> COMMON/MM, LP:, ST: COMMON/PI/-HD = [3,1] COMMON

TKB> /

ENTER OPTIONS:

TKB> PAR = COMMON: 0:16000

TKB> STACK = 0

TKB> 1

5. Logoff

The procedure to execute the D/A modules is as follows:

1. Logon a privileged UIC

2. Execute the indirect command file DA.CMD (i.e., TYPE @ DA)

During the execution, the D/A modules will ask for additional information to set up the D/A operation. It will also set the CPU at the highest hardware priority, i.e., it occupies the CPU. Thus, it will suspend other users' programs and stop the real-time clock. After the D/A operation, it will re-start other users' programs and the real-time clock.

The program listings are as follows:

FORTRAN IV-PLUS V92-51 11:22:38 31-MAR-88 PAGE 2
DASP.FTN /TR:BLOCKS/VR

```

0009      READ(2,112)INSENT,IRATE,NSAMP,IUPPR,ILOVR,NTERMS,H
0010      FORMAT(615,10X,40A1)
0011      WRITE(6,312)INSENT,IRATE,NSAMP,IUPPR,ILOVR,NTERMS,H
0012      FORMAT(1X,615,10X,40A1)
0013      READ(2,111,END=88)IIS
0014      FORMAT(1615)
0015      ERROR CHECKING IN SIZE OF DATA
0016      IF(NSAMP .LE. 16828) GO TO 333
0017      TYPE *, 'YOUR DATA FILE IS LARGER THAN DATA BUFFER I.'
0018      TYPE *, ' # OF SAMPLE ', NSAMP
0019      NSAMP=16828
0020      NF=1
0021      ERROR CHECKING IN MAGNITUDE OF DATA
0022      DO 54 I=1,NSAMP
0023      IF(IIS(I) .GT. 2847) GO TO 334
0024      IF(IIS(I) .LT. -2848) GO TO 334
0025      GO TO 54
0026      NG=NG+1
0027      IF(NG .NE. 1) GO TO 335
0028      NF=1
0029      TYPE *, ' DATA OUTSIDE RANGE OF D/A DATA REGISTER I.'
0030      TYPE *, ' IIS(',I,')=',IIS(I)
0031      CONTINUE
0032      IF(NF .EQ. 8) GO TO 22
0033      TYPE *, ' DO YOU WANT CONTINUE [Y/N] ?'
0034      ACCEPT 336,Q
0035      FORMAT(A1)
0036      IF(Q .EQ. 'N') STOP
0037      NG=8
0038      NF=8
0039      CONDITIONS
0040      TYPE *, ' CHANNEL # ?'
0041      ACCEPT *,MCHNL
0042      NRT=8
0043      OUTPUT DATA TO D/A
0044      HAVE DELAY TIMING CONTROL
0045      CALL DA(NSAMP,MCHNL,NRT)
0046      CONTINUE?
0047      TYPE *, 'DO YOU WANT TO CONTINUE?'
0048      WRITE(5,*) 1=NEW FILE. 2=OUTPUT AGAIN. 3=STOP.'
0049      ACCEPT *,M
0050      IF(M .NE. 1) GO TO 33
0051      CLOSE(UNIT=2)
0052      GO TO 11
0053      IF(M .EQ. 2) GO TO 12
0054      IF(M .EQ. 3) STOP
0055      TYPE *,NA,NB,NC
0056      TYPE *,'?
0057      GO TO 13
0058      END

```


FORTRAN IV-PLUS V02-51 11:23:59 31-MAR-68 PAGE 2
DARAMP.FTN /TR:BLOCKS/VR

```

0011 ACCEPT *.NRT
0012 TYPE *, CHANNEL # ?'
0013 ACCEPT *, MCHNL
0014 RAMP OR SQUARE FUNCTION
0015 TYPE *, RAMP OR SQUARE ? ( 1-RAMP, 2-SQUARE )'
0016 ACCEPT *, NN
0017 IF (NN .NE. 1) GO TO 38
0018 PRODUCE RAMP DATA
0019 IK=8
0020 DO 48 I=1, NSTART, NEND, NSI
0021 IK=IK+1
0022 X=-1. + 1./6488. * (I-1)
0023 X1=2847. * X
0024 X1=X1-B.6
0025 IF (X1 .LT. 8.) GO TO 48
0026 X1=X1+1.
0027 IIS(IK)=X1
0028 GO TO 55
0029 PRODUCE SQUARE DATA
0030 TYPE *, AMPLITUDE ?'
0031 ACCEPT *, NHIGH
0032 IK=8
0033 DO 44 I=1, NSTART, NEND, NSI
0034 IK=IK+1
0035 IIS(IK)=8
0036 IF (IK .GT. 16828) GO TO 488
0037 OUTPUT DATA TO D/A
0038 HAVE DELAY TIMING CONTROL
0039 CALL DA(IK, MCHNL, NRT)
0040 CONTINUE?
0041 TYPE *, 'DO YOU WANT TO CONTINUE?'
0042 WRITE(5, *) 1-NEW PARAMETERS. 2-OUTPUT AGAIN. 3-STOP.'
0043 ACCEPT *, M
0044 IF (M .EQ. 1) GO TO 33
0045 IF (M .EQ. 2) GO TO 12
0046 IF (M .EQ. 3) STOP
0047 TYPE *, NA, NB, NC
0048 TYPE *, ?'
0049 GO TO 13
0050 TYPE *, 'SAMPLE # IS LARGER THAN BUFFER 1'
0051 STOP
0052 END

```


SUB MACRO MILLB 31-MAR-88 11:26 PAGE 1-1

| | | | |
|-----|--------|--------|-----|
| 58 | 000124 | 000248 | NOP |
| 59 | 000126 | 000248 | NOP |
| 60 | 000130 | 000248 | NOP |
| 61 | 000132 | 000248 | NOP |
| 62 | 000134 | 000248 | NOP |
| 63 | 000136 | 000248 | NOP |
| 64 | 000140 | 000248 | NOP |
| 65 | 000142 | 000248 | NOP |
| 66 | 000144 | 000248 | NOP |
| 67 | 000146 | 000248 | NOP |
| 68 | 000150 | 000248 | NOP |
| 69 | 000152 | 000248 | NOP |
| 70 | 000154 | 000248 | NOP |
| 71 | 000156 | 000248 | NOP |
| 72 | 000158 | 000248 | NOP |
| 73 | 000162 | 000248 | NOP |
| 74 | 000164 | 000248 | NOP |
| 75 | 000166 | 000248 | NOP |
| 76 | 000170 | 000248 | NOP |
| 77 | 000172 | 000248 | NOP |
| 78 | 000174 | 000248 | NOP |
| 79 | 000176 | 000248 | NOP |
| 80 | 000200 | 000248 | NOP |
| 81 | 000202 | 000248 | NOP |
| 82 | 000204 | 000248 | NOP |
| 83 | 000206 | 000248 | NOP |
| 84 | 000210 | 000248 | NOP |
| 85 | 000212 | 000248 | NOP |
| 86 | 000214 | 000248 | NOP |
| 87 | 000216 | 000248 | NOP |
| 88 | 000220 | 000248 | NOP |
| 89 | 000222 | 000248 | NOP |
| 90 | 000224 | 000248 | NOP |
| 91 | 000226 | 000248 | NOP |
| 92 | 000230 | 000248 | NOP |
| 93 | 000232 | 000248 | NOP |
| 94 | 000234 | 000248 | NOP |
| 95 | 000236 | 000248 | NOP |
| 96 | 000240 | 000248 | NOP |
| 97 | 000242 | 000248 | NOP |
| 98 | 000244 | 000248 | NOP |
| 99 | 000246 | 000248 | NOP |
| 100 | 000250 | 000248 | NOP |
| 101 | 000252 | 000248 | NOP |
| 102 | 000254 | 000248 | NOP |
| 103 | 000256 | 000248 | NOP |
| 104 | 000260 | 000248 | NOP |
| 105 | 000262 | 000248 | NOP |
| 106 | 000264 | 000248 | NOP |
| 107 | 000266 | 000248 | NOP |
| 108 | 000270 | 000248 | NOP |
| 109 | 000272 | 000248 | NOP |
| 110 | 000274 | 000248 | NOP |
| 111 | 000276 | 000248 | NOP |
| 112 | 000300 | 000248 | NOP |
| 113 | 000302 | 000248 | NOP |
| 114 | 000304 | 000248 | NOP |

```

115 000306 000240
116 000310 000240
117 000312 000240
118 000314 000240
119 000316 000240
120 000320 000240
121 000322 000240
122 000324 000240
123 000326 000240
124 000330 000240
125 000332 000240
126 000334 000240
127 000336 000240
128 000340 000240
129 000342 000240
130 000344 000240
131 000346 000240
132 000350 000240
133 000352 000240
134 000354 000240
135 000356 000240
136 000358 000240
137 000360 000240
138 000362 000240
139 000364 000240
140 000366 000240
141 000368 000240
142 000370 000240
143 000372 000240
144 000374 000240
145 000376 000240
146 000378 000240
147 000380 000240
148 000382 000240
149 000384 000240
150 000386 000240
151 000388 000240
152 000390 000240
153 000392 000240
154 000394 000240
155 000396 000240
156 000398 000240
157 000400 000240
158 000402 000240
159 000404 000240
160 000406 000240
161 000408 000240
162 000410 000240
163 000412 000240

```

```

; END OF DATA?
; SET CHANNEL 0
; RESET D/A
; STARTING ADDRESS OF DATA
; REPEAT
; RESTORE PSW
; TRANSFER FINISHED

```

```

R1 LOOP
R2 00SAMNO,R1
R3 00LDAADR
R4 00LDAADR
R5 00LDAADR
R6 00LDAADR
R7 00LDAADR
R8 00LDAADR
R9 00LDAADR
R10 00LDAADR
R11 00LDAADR
R12 00LDAADR
R13 00LDAADR
R14 00LDAADR
R15 00LDAADR
R16 00LDAADR
R17 00LDAADR
R18 00LDAADR
R19 00LDAADR
R20 00LDAADR
R21 00LDAADR
R22 00LDAADR
R23 00LDAADR
R24 00LDAADR
R25 00LDAADR
R26 00LDAADR
R27 00LDAADR
R28 00LDAADR
R29 00LDAADR
R30 00LDAADR
R31 00LDAADR
R32 00LDAADR
R33 00LDAADR
R34 00LDAADR
R35 00LDAADR
R36 00LDAADR
R37 00LDAADR
R38 00LDAADR
R39 00LDAADR
R40 00LDAADR
R41 00LDAADR
R42 00LDAADR
R43 00LDAADR
R44 00LDAADR
R45 00LDAADR
R46 00LDAADR
R47 00LDAADR
R48 00LDAADR
R49 00LDAADR
R50 00LDAADR
R51 00LDAADR
R52 00LDAADR
R53 00LDAADR
R54 00LDAADR
R55 00LDAADR
R56 00LDAADR
R57 00LDAADR
R58 00LDAADR
R59 00LDAADR
R60 00LDAADR
R61 00LDAADR
R62 00LDAADR
R63 00LDAADR
R64 00LDAADR
R65 00LDAADR
R66 00LDAADR
R67 00LDAADR
R68 00LDAADR
R69 00LDAADR
R70 00LDAADR
R71 00LDAADR
R72 00LDAADR
R73 00LDAADR
R74 00LDAADR
R75 00LDAADR
R76 00LDAADR
R77 00LDAADR
R78 00LDAADR
R79 00LDAADR
R80 00LDAADR
R81 00LDAADR
R82 00LDAADR
R83 00LDAADR
R84 00LDAADR
R85 00LDAADR
R86 00LDAADR
R87 00LDAADR
R88 00LDAADR
R89 00LDAADR
R90 00LDAADR
R91 00LDAADR
R92 00LDAADR
R93 00LDAADR
R94 00LDAADR
R95 00LDAADR
R96 00LDAADR
R97 00LDAADR
R98 00LDAADR
R99 00LDAADR
R100 00LDAADR

```

```

; DATA BUFFER
; PSECT IBUF,D,OVR,GBL
; GLOBL BUF1
; BLKW 16820
; TEMPORARY STORAGE SPACE
; PSECT DATA,D,OVR,GBL
; GLOBL TEMP
; BLKW 1
; TEMP. STORAGE FOR PSW
; INFORMATION TRANSFER
; PSECT INF,D,OVR,GBL
; GLOBL SAMNO,CHANEL,REPEAT
; BLKW 1
; CHANEL: .BLKW 1
; REPEAT: .WORD 0
; DEVICE COMMON BLOCK
; PSECT COMMON,D,OVR,GBL
; BLKW 1
; LDAADR: .BLKW 1
; BLKW 6
; LDADR: .BLKW 1
; BLKW 27
; BLKW 3520
; BLKW 31
; PSW: .BLKW 1
; END

```

```

1 *****
2 "
3 " D/A OUTPUT PROGRAM
4 " SUB PROGRAM: DAB8.MAC
5 "
6 *****
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
10
```

| | | | |
|-----|--------|--------|-----|
| 58 | 000124 | 000240 | NOP |
| 59 | 000126 | 000240 | NOP |
| 60 | 000130 | 000240 | NOP |
| 61 | 000132 | 000240 | NOP |
| 62 | 000134 | 000240 | NOP |
| 63 | 000136 | 000240 | NOP |
| 64 | 000140 | 000240 | NOP |
| 65 | 000142 | 000240 | NOP |
| 66 | 000144 | 000240 | NOP |
| 67 | 000146 | 000240 | NOP |
| 68 | 000150 | 000240 | NOP |
| 69 | 000162 | 000240 | NOP |
| 70 | 000154 | 000240 | NOP |
| 71 | 000156 | 000240 | NOP |
| 72 | 000160 | 000240 | NOP |
| 73 | 000162 | 000240 | NOP |
| 74 | 000164 | 000240 | NOP |
| 75 | 000166 | 000240 | NOP |
| 76 | 000170 | 000240 | NOP |
| 77 | 000172 | 000240 | NOP |
| 78 | 000174 | 000240 | NOP |
| 79 | 000176 | 000240 | NOP |
| 80 | 000200 | 000240 | NOP |
| 81 | 000202 | 000240 | NOP |
| 82 | 000204 | 000240 | NOP |
| 83 | 000206 | 000240 | NOP |
| 84 | 000210 | 000240 | NOP |
| 85 | 000212 | 000240 | NOP |
| 86 | 000214 | 000240 | NOP |
| 87 | 000216 | 000240 | NOP |
| 88 | 000220 | 000240 | NOP |
| 89 | 000222 | 000240 | NOP |
| 90 | 000224 | 000240 | NOP |
| 91 | 000226 | 000240 | NOP |
| 92 | 000230 | 000240 | NOP |
| 93 | 000232 | 000240 | NOP |
| 94 | 000234 | 000240 | NOP |
| 95 | 000236 | 000240 | NOP |
| 96 | 000240 | 000240 | NOP |
| 97 | 000242 | 000240 | NOP |
| 98 | 000244 | 000240 | NOP |
| 99 | 000246 | 000240 | NOP |
| 100 | 000250 | 000240 | NOP |
| 101 | 000252 | 000240 | NOP |
| 102 | 000254 | 000240 | NOP |
| 103 | 000256 | 000240 | NOP |
| 104 | 000260 | 000240 | NOP |
| 105 | 000262 | 000240 | NOP |
| 106 | 000264 | 000240 | NOP |
| 107 | 000266 | 000240 | NOP |
| 108 | 000270 | 000240 | NOP |
| 109 | 000272 | 000240 | NOP |
| 110 | 000274 | 000240 | NOP |
| 111 | 000276 | 000240 | NOP |
| 112 | 000300 | 000301 | DEC |
| 113 | 000302 | 100270 | BPL |
| 114 | 000304 | 013701 | MOV |

RI 1 END OF DATA?
LOOP
00SAMNO,RI

000000'

SUB MACRO M1111 31-MAR-88 11:28 PAGE 1-2

```

115 000310 010437 000002'
116 000314 012737 000000' 000020'
117 000322 012703 000000'
118 000326 005302
119 000330 100256
120 000332 113737 000000' 015776'
121 000340 000207
122
123 000000
124
125 000000
126
127 000000
128
129 000000
130 000000
131 000000
132
133 000000
134 000002
135 000004
136
137 000000
138 000000
139 000002
140 000004
141 000020
142 000022
143 000100
144 015700
145 015776
146

; SET CHANNEL
; RESET D/A
; STARTING ADDRESS OF DATA

R4,00LDAADR
00,00LDADR
00UF1,R3
R2
LOOP
00TEMP,00PSW
PC
; REPEAT
; RESTORE PSW
; TRANSFER FINISHED

; DATA BUFFER
.PSECT IBUF,D,OVR,GBL
.GLOBL BUF1
.BLKV 15820.
BUF1:
; TEMPORARY STORAGE SPACE
.PSECT DATA,D,OVR,GBL
.GLOBL TEMP
TEMP: .BLKV 1 ; TEMP. STORAGE FOR PSW
; INFORMATION TRANSFER
.PSECT INF,D,OVR,GBL
.GLOBL SAMNO,CHANEL,REPEAT
SAMNO: .BLKV 1
CHANEL: .BLKV 1
REPEAT: .WORD 0
; DEVICE COMMON BLOCK
.PSECT COMMON,D,OVR,GBL
LSR: .BLKV 1
LDAADR: .BLKV 1
LOADR: .BLKV 6
.BLKV 27
.BLKV 3620.
.BLKV 31.
PSW: .BLKV 1
.END
000001

```



```

IFB A6 GOTO 98
WAIT FOR
FOR DARAMP=DARAMP
.90: .IFT A2 .GOTO 91
      .SETT A2
WAIT MAC
MAC DA64=DA64
.91: .SETS B1 "DARM64"
      .IFT A7 .GOTO 288
      .SETT A7
      .SETS B2 "DA64"
      .SETS B3 "DARAMP"
      .GOTO 188
.88: .ASK E 8.8KHZ RAMP
      .IFF E .GOTO 388
      .IFT A6 .GOTO 81
      .SETT A6
WAIT FOR
FOR DARAMP=DARAMP
.81: .IFT A4 .GOTO 82
      .SETT A4
WAIT MAC
MAC DAB8=DA88
.82: .SETS B1 "DARM88"
      .IFT A8 .GOTO 288
      .SETT A8
      .SETS B2 "DA88"
      .SETS B3 "DARAMP"
      .OPEN DAB81.CMD
.188: .ENABLE DATA
      .B1' /PR:8-'82'-'83'
COMMON=COMMON:RV
//
.DISABLE DATA
.CLOSE
.WAIT TKB
TKB @DAB81
.288: 1 THE FOLLOWING STEPS HAVE TO BE DONE CAREFULLY.
      1 IF YOU ARE READY, PLEASE TYPE 'RES ...AT.'
      1 OR YOU CAN TYPE 'ABO ...AT.' TO ABORT THIS COMMAND FILE.
.PAUSE DA
INS 'B1'
FIN 'B1'
1 COMMAND WILL BE PAUSED RIGHT HERE.
1 AFTER EXECUTION OF D/A FILE, PLEASE TYPE 'RES ...AT.'
1 TO REMOVE THE TASK.
RUN 'B1'
.PAUSE DA
REM 'B1'
.388: .ASK F DO YOU WANT TO RUN IT AGAIN
      .IFT F .GOTO 2
SET /NOMAIN=COMMON
PIP DAB81.CMD:*/DE
.IFF A1 .GOTO 381
.WAIT PIP
PIP DASP.OBJ:*/DE
.381: .IFT A2 .GOTO 382
.WAIT PIP
PIP DA64.OBJ:*/DE
.382: .IFT A3 .GOTO 383
.WAIT PIP
PIP DAB84 TCV:*/DE

```

```
.393: .IFF A4 .GOTO 384
.WAIT PIP
PIP DARG.ORG;"/DE
.394: .IFF A5 .GOTO 385
.WAIT PIP
PIP DARG.ORG.TSK;"/DE
.395: .IFF A6 .GOTO 386
.WAIT PIP
PIP DARG.ORG;"/DE
.396: .IFF A7 .GOTO 387
.WAIT PIP
PIP DARG.ORG.TSK;"/DE
.397: .IFF A8 .GOTO 388
.WAIT PIP
PIP DARG.ORG.TSK;"/DE
.398: ; END.
```

END

DATE
FILMED

8-80

DTIC